

**INTERPOLATIVE DECOMPOSITION AND ITS
APPLICATIONS**

NG HAO TIAN

UNIVERSITI TUNKU ABDUL RAHMAN

INTERPOLATIVE DECOMPOSITION AND ITS APPLICATIONS

NG HAO TIAN

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Science (Honours)
Applied Mathematics with Computing**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

September 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : Hao Tian

Name : Ng Hao Tian


ID No. : 2106608

Date : 06 September 2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**INTERPOLATIVE DECOMPOSITION AND ITS APPLICATIONS**” was prepared by **NG HAO TIAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Applied Mathematics with Computing at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Ng Wei Shean

Date : 6 September 2024

Signature : -

Co-Supervisor : -

Date : -

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2024, Ng Hao Tian. All right reserved.

ABSTRACT

Matrix decomposition is extremely useful for data compression and efficient storage. As images are increasingly used across the Internet, the challenge of managing large file sizes becomes more significant. However, higher-quality images often come with large file sizes, making transmission extremely cumbersome. Matrix decomposition techniques can address this issue by compressing images to smaller file sizes while preserving much of the image fidelity. Although Singular Value Decomposition (SVD) has been widely used, Interpolative Decomposition (ID) has recently emerged as an alternative. This report explores the applicability of ID in image processing, particularly in image compression and compares its performance with that of SVD.

We used four different images with unique characteristics to evaluate the performance of the matrix decomposition methods. Based on this, we deduced general compression parameters for both methods and tested these parameters using a separate image. All images were taken from the USC-SIPI Image Database. Our analysis showed that both ID and SVD struggled to compress images with darker regions. Nevertheless, ID generally produced better image quality, while SVD was more effective in reducing file size. For 512×512 pixel images, we established the following generalised compression parameters: for grayscale images, $k = 243$ with a threshold of 0.01 for ID, and $k = 89$ for SVD; for colour images, $k = 370$ with a threshold of 0.01 for ID, and $k = 184$ for SVD.

TABLE OF CONTENTS

DECLARATION		i
APPROVAL FOR SUBMISSION		ii
ABSTRACT		iv
TABLE OF CONTENTS		v
LIST OF TABLES		vii
LIST OF FIGURES		ix
CHAPTER		
1	INTRODUCTION	1
1.1	General Introduction	1
1.2	Motivation	2
1.3	Problem Statement	2
1.4	Objectives	3
1.5	Project Schedule Timeline	3
2	LITERATURE REVIEW	6
2.1	Introduction	6
2.2	Definitions	6
2.3	Interpolative Decomposition (ID)	6
2.3.1	Proposed Algorithms	7
2.4	Singular Value Decomposition (SVD)	8
2.5	Non-Negative Matrix Decomposition (NMF)	9
2.6	QR Decomposition	10
2.7	Strength of Each Matrix Decomposition	10
2.8	Application of Matrix Decomposition	13
2.8.1	Application in Quantum Mechanics	13
2.8.2	Application in Signal Processing	13
2.8.3	Application in Image Processing	13
3	METHODOLOGY AND WORK PLAN	16
3.1	Introduction	16

3.2	Data Understanding	16
3.3	Image Preprocessing	17
3.4	Image Compression	17
3.5	Performance Metrics	18
4	RESULTS AND DISCUSSION	20
4.1	Introduction	20
4.2	Initial Findings	20
4.3	Finding Optimal Parameters	24
	4.3.1 Optimal Parameters for ID	24
	4.3.2 Optimal Parameters for SVD	36
4.4	Comparative Analysis of ID-method and SVD- method	44
5	CONCLUSIONS AND RECOMMENDATIONS	48
5.1	Conclusions	48
5.2	Recommendations for future work	49
	REFERENCES	50
	APPENDICES	53

LIST OF TABLES

Table 1.1:	Gantt Chart Final Year Project I (January Trimester 2024).	3
Table 1.2:	Gantt Chart Final Year Project II (May Trimester 2024).	4
Table 4.1:	Table of k and their respective reconstruction errors and compression ratios.	23
Table 4.2:	The best configuration results for the Baboon image using ID.	25
Table 4.3:	The best configuration results for the Sailboat on Lake image using ID.	26
Table 4.4:	The best configuration results for the Peppers image using ID.	27
Table 4.5:	The best configuration results for the Airplane F-16 image using ID.	28
Table 4.6:	The best configuration results for the Airplane F-16 image using ID with a threshold value of 0.01.	28
Table 4.7:	Compression results for the four images using ID with $k_{avg} = 243$ and a threshold of 0.01.	29
Table 4.8:	The best configuration results for the four images using ID.	31
Table 4.9:	Compression results for the four images using ID with $k_{avg} = 370$ and a threshold of 0.01.	32
Table 4.10:	Comparison results between compressing using optimal and generalised parameters on both grayscale and colour images using ID.	34
Table 4.11:	The best configuration results for the four images using SVD.	37
Table 4.12:	Compression results for the four images using SVD with $k_{avg} = 89$.	38
Table 4.13:	The best configuration results for the four colour images using SVD.	40
Table 4.14:	Compression results for the four images using SVD with $k_{avg} = 184$.	42

Table 4.15:	Comparison results between compressing using optimal and the general parameter for both grayscale and colour images using SVD.	43
Table 4.16:	Summary of the similarities and differences between ID and SVD method.	47

LIST OF FIGURES

Figure 2.1:	Illustration of ID.	7
Figure 3.1:	Flowchart to obtain the optimal compression parameters of an image using ID.	16
Figure 3.2:	Python code snippet to modify the interpolation matrix, Z .	18
Figure 3.3:	Python code snippet to calculate reconstruction error.	19
Figure 4.1:	Uncompressed Baboon image.	20
Figure 4.2:	MSE vs k with a threshold of 0.01.	21
Figure 4.3:	PSNR vs k with a threshold of 0.01.	21
Figure 4.4:	SSIM vs k with a threshold of 0.01.	22
Figure 4.5:	ID compression of different k values with a threshold of 0.01.	23
Figure 4.6:	Compressed Baboon images with their SSIM values.	25
Figure 4.7:	Comparison between uncompressed and compressed Baboon images with optimal compression parameters.	26
Figure 4.8:	Comparison between uncompressed and compressed Sailboat on Lake images with optimal compression parameters.	27
Figure 4.9:	Comparison between uncompressed and compressed Peppers images with optimal compression parameters.	27
Figure 4.10:	Comparison of three images: the uncompressed Airplane F-16 image, the compressed image using the optimal parameters, and the compressed image with a threshold value of 0.01.	29
Figure 4.11:	Comparative images of their uncompressed and recompressed versions with $k_{avg} = 243$ and a threshold of 0.01.	30
Figure 4.12:	Comparative images of uncompressed and optimally compressed versions.	31
Figure 4.13:	Comparative images of their uncompressed and recompressed versions with $k_{avg} = 370$ and a threshold of 0.01.	33

Figure 4.14:	Comparative images of uncompressed, optimally compressed, and generally compressed versions.	35
Figure 4.15:	Graphs of SSIM and Compression Ratio against k .	36
Figure 4.16:	Comparison of the four images using their best configurations versus $k_{avg} = 89$, alongside their uncompressed versions.	39
Figure 4.17:	Comparative images of uncompressed and optimally compressed versions.	41
Figure 4.18:	Comparative images of their uncompressed and recompressed versions with $k_{avg} = 184$.	42
Figure 4.19:	Comparative images of uncompressed, optimally compressed, and generally compressed versions.	44
Figure 4.20:	Comparison of SSIM for ID and SVD using the generalised parameters.	45
Figure 4.21:	Comparison of SSIM for ID and SVD using the generalised parameters.	46

CHAPTER 1

INTRODUCTION

1.1 General Introduction

As technology continues to evolve, taking high-quality pictures is not as challenging as it can be done with a smartphone. However, transmitting these images can be cumbersome due to their high resolution and larger file sizes. Moreover, often the content is much more significant than the image quality. Therefore, an image compression approach is required to reduce file sizes for convenient transmission. Matrix decomposition is one technique for image compression. By breaking down complex matrices into simpler components, matrix decomposition offers insights into the underlying structure of the data, which is beneficial for various computational tasks and analysis.

One such classical method is the Lower-Upper (LU) factorisation, which efficiently solves systems of linear equations and calculates determinants (Mittal and Al-Kurdi, 2002; Su et al., 2018). Singular value decomposition (SVD) is another well-known method that is widely used in a variety of fields, including collaborative filtering, image processing, and data compression. Additionally, QR decomposition decomposes a matrix into the product of Q, an orthogonal matrix and R, an upper triangular matrix. Because of its stability and robustness, QR decomposition is widely used in data analysis, optimisation, and numerical computations. It provides reliable answers for eigenvalue calculations, linear least squares issues, and orthogonalisation processes (Kawamura and Suda, 2021).

In recent years, Interpolative Decomposition (ID) has become a powerful approach for matrix compression and approximation. ID focuses on finding a subset of columns that represent the fundamental composition of the original matrix, known as the skeleton. ID creates a low-rank approximation by interpolating across chosen columns, which significantly lowers computational complexity and memory requirements while preserving important properties.

These various matrix decomposition techniques present different trade-offs in terms of accuracy, computational complexity, and suitability for particular problem fields. Understanding their concepts and capabilities is

critical in image processing, whether it involves compression, denoising, or other transformational operations.

1.2 Motivation

The volume of data being generated grows at an exponential rate. Knowledge is hidden within these data and is waiting to be discovered. Data analysts apply data mining or predictive modelling techniques to uncover the secrets hidden within them. However, as the volume of information grows, the curse of dimensionality is also experienced. A suitable number of attributes is mostly enough to produce the required knowledge, too many of them may cause disturbances in the decision-making process or impede the calculation process. To address this challenge, matrix decomposition or matrix factorisation techniques may be used to decompose a vast matrix into its constituent parts to leverage the computation process. By leveraging interpolative decomposition methods, we can effectively navigate the complexities of high-dimensional data and uncover actionable insights that drive informed decision-making and enhance overall data-driven strategies.

1.3 Problem Statement

Matrices are widely used for representations of numerical data in various fields, from systems of equations to image processing. However, manipulating a large matrix may be tedious to operate with, thus, techniques such as LU decomposition, QR decomposition, SVD, and ID are invented to help break down the large matrix into a simpler form to work on.

ID is a numerical approach for approximating large matrices with low-rank matrices. The skeleton matrix is usually the matrix to represent the original matrix. This project aims to explore the practical applications of ID in image processing.

Furthermore, the project aims to compare ID against other matrix decomposition techniques to assess its performance and applicability for various applications. By examining the strengths and weaknesses of ID in comparison with alternative methods, this study aims to provide insights into its optimal usage scenarios and potential improvements.

1.4 Objectives

The objectives of this project are summarised as follows:

- (1) Explore the applicability of Interpolative Decomposition (ID) in image processing by utilising the properties of ID.
- (2) Conduct and compare the performance and suitability of different matrix decomposition techniques including ID and Singular Value Decomposition (SVD), in image processing.
- (3) Obtain a generalised compression parameters to compress images using ID and SVD.

1.5 Project Schedule Timeline

The entire project spans two trimesters, consisting of Project I and Project II. Table 1.1 and Table 1.2 show the Gantt Charts for Project I and Project II respectively.

Table 1.1: Gantt Chart Final Year Project I (January Trimester 2024).

Week												
1	2	3	4	5	6	7	8	9	10	11	12	13

Task	
	Title Registration
	Submission of Biweekly Report
	Sourcing ID and Image Processing Related Material
	Writing Proposal
	Submission of Project Proposal
	Writing Algorithm and Test for Grayscale Images
	Writing Interim Report
	Submission of Interim Report
	Oral Presentation

Table 1.2: Gantt Chart Final Year Project II (May Trimester 2024).

Week												
1	2	3	4	5	6	7	8	9	10	11	12	13

Task	
	Expanding the Report from Project I
	Sourcing ID and Image Processing Related Material
	Writing Algorithm and Test for RGB Images
	Refining the Algorithm
	Writing Final Report
	Submission Mid-Semester Monitoring Form
	Prepare FYP Poster
	Submission of Poster
	Submission of Presentation Slide
	Submission of All Relevant Documents
	Oral Presentation

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Low-rank approximation is a popular technique used in data analysis to reduce the memory footprint and computational complexity (Muravev et al., 2018). One common method for this is SVD, but ID is also emerging as a promising alternative.

2.2 Definitions

Some common definitions are listed below.

Let i, j be two positive integers where $1 \leq i \leq m, 1 \leq j \leq n$ and let $A = (a_{i,j}) \in \mathbb{R}^{m \times n}$.

- (1) A is a diagonal matrix when $a_{i,j} = 0$ for all $i \neq j$.
- (2) A is a lower triangular matrix when $a_{i,j} = 0$ for all $i < j$.
- (3) A is an upper triangular matrix when $a_{i,j} = 0$ for all $i > j$.
- (4) A is a lower diagonal matrix when $a_{i,j} = 0$ for all $i < j$ and $a_{i,j} \neq 0$.
- (5) A is an upper diagonal matrix when $a_{i,j} = 0$ for all $i > j$ and $a_{i,j} \neq 0$.
- (6) $A = I_n = I \in \mathbb{R}^{n \times n}$ is the identity matrix of order n when $a_{i,j} = 0$ for all $i \neq j$ and $a_{i,j} = 1$ for all $i = j$.

2.3 Interpolative Decomposition (ID)

This subsection summarises the definition and properties of ID. The following lemma defines the interpolative decomposition.

Lemma 1 (Liberty et al., 2007): Suppose m, n are two positive integers, and A is an $m \times n$ matrix. Then, for any real number k with $k \leq m$ and $k \leq n$, there exists an $m \times k$ skeleton matrix C whose columns are a selected subset of the columns of the matrix A , and a $k \times n$ interpolation matrix Z , such that

1. An identity matrix of order k , I_k is a subset of the columns of the matrix Z ,
2. Every element of Z will have a magnitude less than or equal to a small positive integer, say 2,
3. The k th greatest singular value of Z is at least 1,
4. $\|Z_{k \times n}\| \leq \sqrt{4k(n-k) + 1}$,
5. When $k = n$ or $k = m$, $A = CZ$, and
6. When $k < n$ or $k < m$, $\|C_{m \times k}Z_{k \times n} - A_{m \times n}\|_2 \leq \sqrt{k(n-k) + 1}\sigma_{k+1}$

where σ_{k+1} is the $(k + 1)$ st greatest singular value of A .

Property 2 ensures that the decomposition is numerically stable. Property 3 follows directly from property 1. Combining properties 1 and 2 yields property 4. Property 5 is the interpolative decomposition of A , whereas property 6 is the k -rank approximation of A . (Liberty et al., 2007)

Figure 2.1 is the illustration of ID where A is the original matrix and after applying ID, it will decompose the matrix to a skeleton matrix C and an interpolation matrix Z .

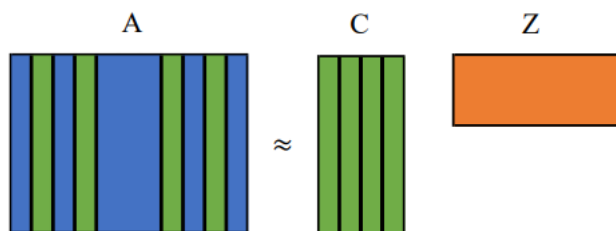


Figure 2.1: Illustration of ID.

2.3.1 Proposed Algorithms

ID is calculated by modifying pivoted QR, as proposed by Golub (1965). The computation cost of ID is $\mathcal{O}(nmk)$, which is cheaper than SVD ($\mathcal{O}(nm \min(n, m))$), where n, m, k are the number of rows, the number of columns, and the rank of original matrix, respectively. As $k \leq \min(n, m)$, ID is particularly useful when solving rank-deficient least squares problems.

Many algorithms have been proposed to solve the interpolative decomposition problem such as (Woolfe et al., 2008), (Liu and He, 2019), (Bhaskara et al., 2020), (Advani and O’Hagan, 2022), etc. Python’s called `scipy.linalg.interpolative` library offers two methods for calculating ID: a deterministic approach (SciPy ID) and a randomised approach (SciPy RID). Advani, Crim and O’Hagan (2020), and Advani and O’Hagan (2022) proposed two simplified algorithms for computing ID in Python. The deterministic algorithm, called Optim ID, selects columns of C through column-pivoted QR decomposition and computes Z via least-squares. On the other hand, the randomised algorithm, Optim RID, speeds up the calculation by using column sampling by sacrificing some accuracy, particularly in very sparse datasets.

Both Optim ID and Optim RID algorithms outperform existing methods in the SciPy library by demonstrating greater computational efficiency and accuracy in constructing ID. Specifically, Optim ID computes approximations faster than SciPy ID while maintaining similar accuracy, and Optim RID consistently shows better performance than SciPy RID across most datasets, particularly on dense datasets like MNIST and Fashion-MNIST.

2.4 Singular Value Decomposition (SVD)

Let A be an $m \times n$ matrix, SVD decomposes A into a product of three matrices,

$$A = U\Sigma V^T.$$

Here,

- U is an $m \times m$ orthogonal matrix, where its columns represent the left singular vectors,
- Σ is an $m \times n$ diagonal matrix with decreasing singular values, and
- V^T is an $n \times n$ orthogonal matrix, whose rows represent the right singular vectors.

Both U and V^T are unitary matrices, thus, $U^T = U^{-1}$ and $V^T = V^{-1}$. The singular values in Σ indicate the importance of the corresponding singular vectors in the decomposition, with higher values indicating greater importance (Strang, 2006). Another type of SVD, called the Truncated SVD, is similar to

the classic SVD with the difference that only the top k significant singular values are considered, where k is a user-determined integer parameter. The choice of k plays a crucial role in determining the level of approximation and compression achieved.

With a higher k value, more singular values and singular vectors from the original matrix are retained, which increases the accuracy of the approximation and reduces matrix norms (e.g., Frobenius norm or spectral norm). As a result, the compression ratio is lower to store more information of the original data and the compressed representation is less compact.

SVD and its variants are commonly used to perform data compression (Bentbib, Kreit and Labaali, 2022), dimensionality reduction (Libal, Baras and Johansson, 2020), and data reconstruction (Intawichai and Chaturantabut, 2022).

2.5 Non-Negative Matrix Decomposition (NMF)

Suppose an $m \times n$ matrix A , whose elements are all non-negative. NMF aims to approximate A into $A \approx WH$, where $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ are both nonnegative matrices and k is the desired low rank. This is an optimisation problem where it minimises the Frobenius norm

$$\min_{W \geq 0, H \geq 0} \|A - WH\|_F^2.$$

This optimisation problem can be solved by gradient descent (Lin, 2007) or the multiplicative update rule (Lee and Seung, 2000). Each column of W can be interpreted as a basis vector or a feature. These basis vectors capture underlying patterns or components present in the data. Each column of H represents the coefficients or weights to reconstruct the original matrix A . The i^{th} column of A , \mathbf{a}_i is a linear combination of the columns of W weighted by the coefficients in \mathbf{h}_i (i^{th} column of H)

$$\mathbf{a}_i \approx W\mathbf{h}_i.$$

2.6 QR Decomposition

Given a matrix $A \in \mathbb{R}^{m \times n}$, QR decomposition decomposes A into an orthonormal matrix, Q , and an upper triangular matrix, R , such that

$$A = QR.$$

If A is a full rank matrix, i.e. all columns are linearly independent, then R will be an upper triangular matrix. The orthonormality of the columns of Q implies that each column in Q is of unit length 1 and every pair of columns in Q are orthogonal (perpendicular) to each other. In addition, any column vector in the column space of A can be formed by performing linear combinations on columns of Q . On the other hand, R is the scaling factor of Q to form A .

The application of QR decomposition includes solving linear least squares problems, solving generalised linear regression model problems, and finding the eigenvalue and eigenvector of a matrix (Anderson, Bai and Dongarra, 1992).

2.7 Strength of Each Matrix Decomposition

ID's strength compared to other matrix decompositions is that it reuses the columns of the original matrix. Thus, it can save computational resources, both in terms of time and space. Another strength of ID is that it can inherit the structure of the columns of the original matrix. This includes sparsity, locality, and factorised form (Ying, et al., 2018). Keeping the structure of the original matrix is extremely important, as it helps maintain data integrity, increases reconstruction accuracy, and preserves its features.

SVD is considered one of the most numerically stable methods for matrix decomposition (Golub and Van Loan, 2013). The stability of SVD helps ensure that small changes in the input matrix result in small changes in the output, thereby maintaining the integrity of the solution. Another strength of SVD is the guarantee of decomposing any matrix A into a specific form: $A = U\Sigma V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix containing singular values. This universality means that SVD can be applied to

matrices of any shape (rectangular, square, etc.), making it a highly versatile tool in linear algebra (Lay, Lay and McDonald, 2014).

One of the key strengths of NMF is its ability to decompose a non-negative matrix into the product of two non-negative matrices, thereby revealing its underlying structure and features while preserving non-negativity in the factorised components. This property makes NMF particularly suited for interpretability in applications such as image processing and text mining, as the original data are non-negative (Zhang, 2024).

One classical advantage of QR factorisation is its ability to solve least squares problems. Given a tall matrix $A_{m \times n}$ with $m > n$, the least squares problem is to solve $Ax = b$. This problem can be approached in the following three ways:

- (1) By solving the normal equation $A^T Ax = A^T b$.
- (2) By solving the equation $Rx = Q^T b$, where $A = QR$ is the QR factorisation of A .
- (3) By solving $x = V\Sigma^{-1}U^T b$, where $A = U\Sigma V^T$ is the SVD of A .

Method 1 is the simplest and fastest approach to solve the problem. However, it is less stable, as the condition number is squared.

Proof.

The condition number of a matrix M is defined as:

$$\kappa(M) = \|M^{-1}\| \|M\|.$$

For a matrix A the condition number is:

$$\kappa(A) = \frac{\sigma_{max}}{\sigma_{min}},$$

where σ_{max} and σ_{min} is the largest and smallest singular value of A .

According to the SVD decomposition of A , we have $A = U\Sigma V^T$. Computing $A^T A$ using SVD, we get:

$$A^T A = (U\Sigma V^T)^T (U\Sigma V^T)$$

$$\begin{aligned}
&= V\Sigma^T U^T U \Sigma V^T \\
&= V\Sigma^T \Sigma V^T.
\end{aligned}$$

Note that $U^T = U^{-1}$ implies $U^T U = I$.

Here $\Sigma^T \Sigma$ is the diagonal matrix where each diagonal entry is the square of the corresponding singular value of A . Specifically, if $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$, where $r = \min(m, n)$, then:

$$\Sigma^T \Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_r^2).$$

To find the condition number for method 1, we compute $\kappa(A^T A)$:

$$\kappa(A^T A) = \|(A^T A)^{-1}\| \|A^T A\|.$$

As previously shown, the largest and smallest singular value of $A^T A$ are σ_{max}^2 and σ_{min}^2 , respectively. Thus, we have:

$$\begin{aligned}
\kappa(A^T A) &= \frac{\sigma_{max}^2}{\sigma_{min}^2} \\
&= \left(\frac{\sigma_{max}}{\sigma_{min}} \right)^2 \\
&= \kappa(A)^2
\end{aligned}$$

This shows that the condition number is squared when solving the least squares problem through the normal equation. ■

Considering both stability and speed, method 2 (solving through the QR factorisation) is the optimal choice, with a computational cost $\mathcal{O}(2mn^2)$ (if using the Gram-Schmidt algorithm), which is cheaper than the SVD method (method 3) of computational cost $\mathcal{O}(2mn^2 + 11n^3)$ (Torberg, 2013). Method 3 is more appropriate when A is rank-deficient.

2.8 Application of Matrix Decomposition

This section exemplifies some applications of how matrix decomposition is used in various fields.

2.8.1 Application in Quantum Mechanics

The usage of ID is also present in the realm of quantum mechanics, particularly in addressing nonlinear eigenvalue problems within Kohn-Sham density function theory (DFT). In this context, Damle, Lin and Ying (2017) used the Selected Columns of Density Matrix (SCDM) approach with ID to efficiently approximate solutions by focusing on localised and sparse columns of the density matrix. Another application is the electron repulsion integral tensor, in which ID is used in conjunction with Fast Fourier Transforms (FFTs) to compress the low-rank matrix representing basis functions in electron systems. This demonstrates the effectiveness of ID in reducing computational complexity and increasing the efficiency of quantum mechanical simulations (Lu and Ying, 2015).

2.8.2 Application in Signal Processing

Tang, Ng and Liew (2023) presented insights into the use of ID in signal processing, specifically in the context of musical instrument source separation. The study compared the performance of ID, Non-negative Matrix Factorisation (NMF), and Convolutional Non-negative Matrix Factorisation (CNMF) in separating musical instrument sources. As opposed to expectation, the results showed that ID did not perform as effectively as NMF and CNMF. The authors regarded this mismatch as the inherent nature of ID, which involves randomly selecting columns from the original matrix, potentially resulting in redundancy and unwanted data in the factorised matrix. This observation highlights the importance of understanding the underlying mechanisms of ID and its implications in different problem domains.

2.8.3 Application in Image Processing

There are two types of image compression: lossy and lossless. Lossless compression is a technique that reduces a file to a smaller size while allowing it to be reconstructed without loss of information. Lossless compression methods

include Variable Length Coding, Run Length Encoding, Differential Coding and Predictive Coding (Sahu and Satao, 2016).

Lossy compression, on the other hand, is a compression technique that reduces the file size of an image by deleting certain features that are nearly unnoticeable to the human eye. This process is irreversible; thus, the original image cannot be fully restored from the compressed version. Methods such as Quantisation, Transform Coding, Fractal Coding and Wavelet Coding are commonly used for lossy compression (Sahu and Satao, 2016). Lossy compression is employed in this project.

Kiran et al. (2023) applied SVD to compress images by utilising logic similar to Truncated SVD. Rather than truncating the matrix, they set the truncated part to zero. According to their findings, the image is efficiently compressed since less storage space is needed to record matrix information at lower matrix ranks.

In contrast, Varghese and Saroja (2021) compressed an image by applying SVD on a hexagonal grid image. Mersereau (1979) pointed out that to achieve similar outcomes, a hexagonal lattice structure needs just 13.4% fewer sample points than a conventional square lattice structure, which means lesser storage and computational resources are needed. Additionally, the similarity of hexagonal patterns to photoreceptors in the human retina served as inspiration for the hexagonal grid image processing method. Varghese and Saroja (2021) first transformed a square lattice image into a pseudo-hexagonal structure based on the design by Wüthrich and Stucki (1991). After approximating the image using a reasonable rank- k approximation, they regenerated the matrix by eliminating singularities in the Σ matrix. This study demonstrates how SVD-based image compression performs better in the hexagonal domain than it does in the square domain.

Kong et al. (2017) proposed an image compression scheme in Wireless Multimedia Sensor Networks (WMSNs) using Non-Negative Matrix Decomposition (NMF). While the primary goal of their scheme is not solely image compression, the authors employed a notable method for this purpose. They observed that the NMF algorithm requires more computational resources to achieve higher image quality, resulting in a longer compression time for larger images. To address this, the authors proposed an adaptive blocking image

compression mechanism. This method partitions an image into smaller blocks, claiming that it can achieve a higher compression ratio and better image restoration with fewer iterations. Their conclusion indicates that the blocking method outperforms SVD in terms of image restoration, given the same blocking size and compression ratio.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This chapter introduces the project's flow to achieve the targeted objectives. This project revolves around Interpolative Decomposition (ID) and its application in image processing. Figure 3.1 shows the flowchart to obtain the optimal compression parameters an image using ID.

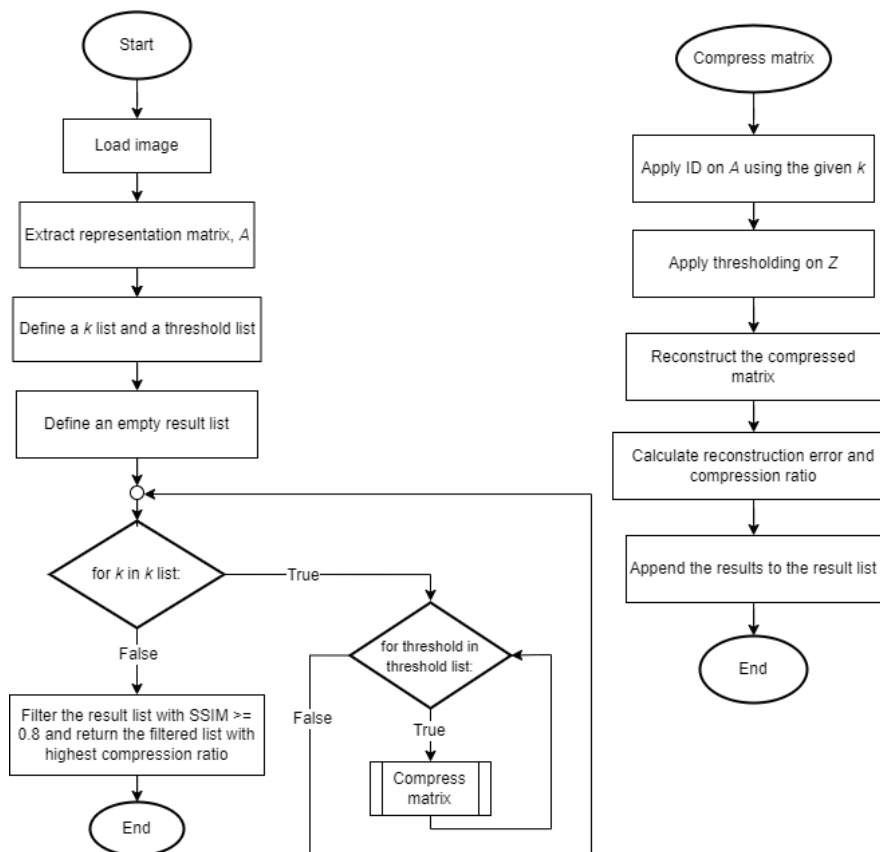


Figure 3.1: Flowchart to obtain the optimal compression parameters of an image using ID.

3.2 Data Understanding

Initially, an image is loaded into Python through Python libraries such as OpenCV (cv2) or Pillow (PIL). The loaded image is essentially a collection of pixels arranged in a grid format and each pixel has a value representing its colour intensity. For grayscale images, the matrix is a two-dimensional matrix

with each entry containing a value from 0 (black) up to 255 (white) with shades of grey in between, representing the colour intensity of each pixel. For colour images (RGB), the matrix has three channels (red, followed by green, then blue), each channel with its two-dimensional array storing different colour intensities for each pixel. Converting the image into a matrix creates a mathematical representation of the image. This, allows us to obtain information on colour intensity in each pixel, thus, enable for various image-processing tasks.

3.3 Image Preprocessing

Once the image is loaded, it undergoes an initial conversion to a grayscale format, and its representation matrix, denoted as A , is extracted. Assuming without loss of generality, that the number of rows, m is less than or equal to the number of columns, n , if this condition is not met, the transpose of the matrix is taken. Once, the image is ready, it can proceed to the next step, compression.

3.4 Image Compression

An iterative process is employed to select optimal parameters (k and threshold). Following this, the ID process is applied to A , yielding an approximation:

$$A \approx CZ,$$

where C is an $m \times k$ matrix, comprising columns selected from A , and Z is the interpolation matrix, a $k \times n$ matrix. The parameter k represents the rank of the desired low-rank approximation of A .

In the following step, the interpolation matrix is modified such that any entries below a certain threshold are set to the minimum value of Z . The value of the threshold is determined through experimentation to ensure that the final image is visually acceptable and effectively reduces the image size. To modify the interpolation matrix, a Python code snippet in Figure 3.2 is used. The code sets any values with an absolute value less than the specified threshold to Z 's minimum after the interpolation matrix, Z is normalised using a min-max scaler.

```

1 # Apply threshold for compression
2 # Normalise the entries of interpolation matrix, Z
3 maximum = numpy.max(Z) # maximum value of the matrix
4 minimum = numpy.min(Z) # minimum value of the matrix
5 normalised_Z = (Z - minimum) / (maximum - minimum)
6
7 # Any entries less than the threshold will be converted to its minimum
8 Z[normalised_Z < threshold] = minimum

```

Figure 3.2: Python code snippet to modify the interpolation matrix, Z .

This adjustment is because each entry in the interpolation matrix represents the weights and scaling factors used in reconstructing the original matrix from the selected subset. This approach assumes that lower-value elements contribute less to reconstruction accuracy and can be discarded without significantly impacting the final image quality. Then the compressed matrix, \tilde{A} is reconstructed using the new interpolated matrix, Z' .

$$CZ \approx \tilde{A} = CZ'.$$

3.5 Performance Metrics

The compression error, denoted as ϵ_c , of the compressed matrix can be calculated using the Frobenius norm,

$$\|A - \tilde{A}\|_F^2 = \epsilon_c.$$

Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) are the two metrics used to evaluate the quality of image reconstruction tasks. MSE quantifies the average squared difference between corresponding pixels in the original image and the reconstructed image after compression. A lower MSE value indicates a smaller overall difference and potentially better reconstruction quality. The formula for MSE is:

$$MSE = \frac{1}{m \times n} \sum_{i=1}^m \sum_{j=1}^n (\tilde{A}_{i,j} - A_{i,j})^2.$$

In contrast, PSNR provides a measure of the signal-to-noise ratio in decibels (dB). A higher PSNR value signifies a higher ratio of the original signal

power (image content) to the noise power (introduced by compression artefacts). The formula for PSNR is:

$$PSNR = 10 \times \log_{10} \frac{(2^N - 1)^2}{MSE},$$

where N is the bit depth per channel of the image.

Figure 3.3 illustrates the Python code used to calculate the reconstruction error of an image. The Structural Similarity Index Measure (SSIM) is another performance metric used to assess image quality.

```

1  def calc_reconstruction_error(original_image, compressed_image):
2      # Ensure images are of the same size
3      assert original_image.shape == compressed_image.shape, \
4          "Images must have the same shape"
5
6      # Calculate MSE
7      mse = numpy.mean((original_image - compressed_image) ** 2)
8
9      # Calculate PSNR
10     max_intensity = 255.0 # Assuming 8-bit image data
11     if mse == 0:
12         psnr = float('inf') # Perfect match
13     else:
14         psnr = 10 * numpy.log10((max_intensity**2) / mse)
15
16     # Calculate SSIM
17     data_range = original_image.max() - original_image.min()
18     ssim_index = ssim(original_image, compressed_image,
19                       data_range=data_range)
20     return mse, psnr, ssim_index

```

Figure 3.3: Python code snippet to calculate reconstruction error.

These steps are iteratively repeated until suitable parameters are identified. The process is then adapted for RGB format images, with slight modifications in error computation.

The performance of ID in image compression is compared with SVD by compressing the same set of images.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, we present the results of our study on Interpolative Decomposition (ID) and its application in image compression. Our objective was to explore the applicability of ID for compressing images by using its properties. The following sections detail the process of identifying optimal parameters for compressing 8-bit images using ID. All images analysed in this project were obtained from the USC-SIPI Image Database.

4.2 Initial Findings

Before determining the optimal parameters to compress an image using ID, it is crucial to understand the relationship between the k value, the rank of the low-rank approximation, and the reconstruction metrics. For this demonstration, we used the grayscale image, Baboon, which has a size of 512×512 pixels, as shown in Figure 4.1.

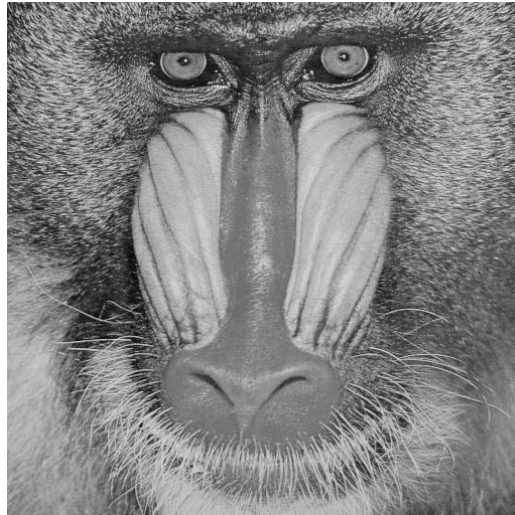


Figure 4.1: Uncompressed Baboon image.

Different values of k were used to decompose a grayscale image, and each resulting metric was evaluated.

To ensure a fair comparison, a constant threshold value, of 0.01 was applied when plotting each metric against the k value. Figure 4.2 illustrates that the Mean Square Error (MSE) decreases as the value of k increases.

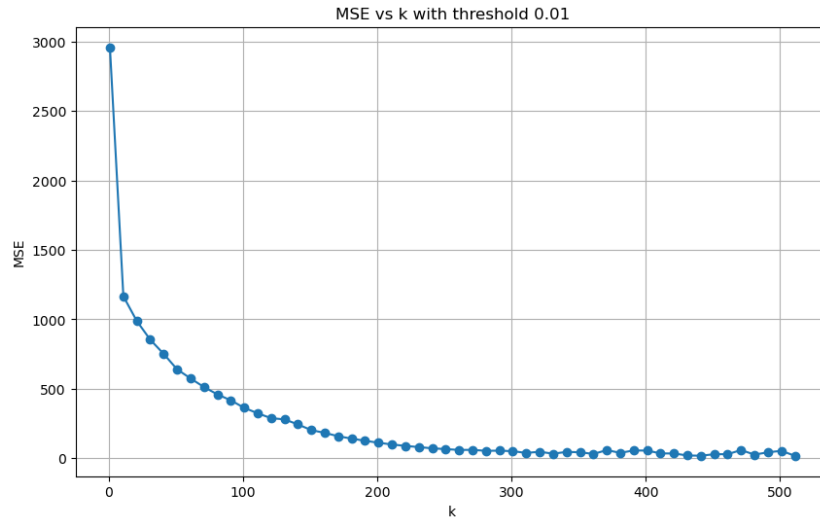


Figure 4.2: MSE vs k with a threshold of 0.01.

Figure 4.3 and Figure 4.4, depict the relationships between the Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index Measure (SSIM) respectively with different k values. These graphs demonstrate that both PSNR and SSIM increase as the k value rises.

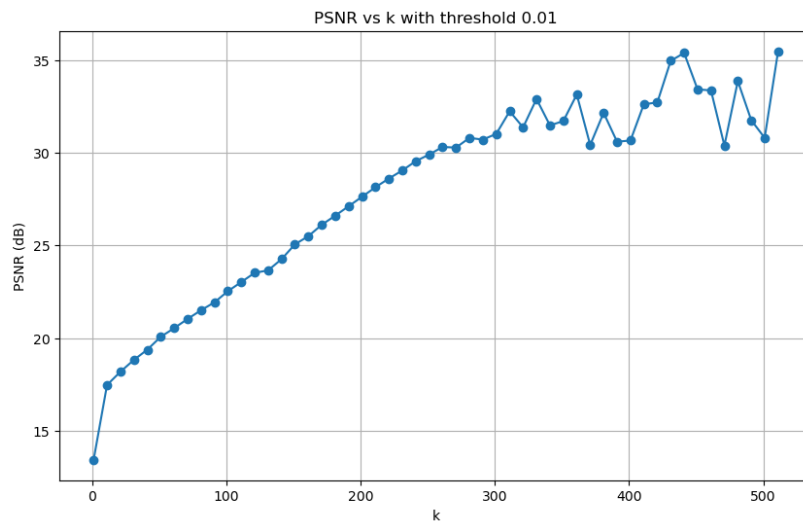


Figure 4.3: PSNR vs k with a threshold of 0.01.

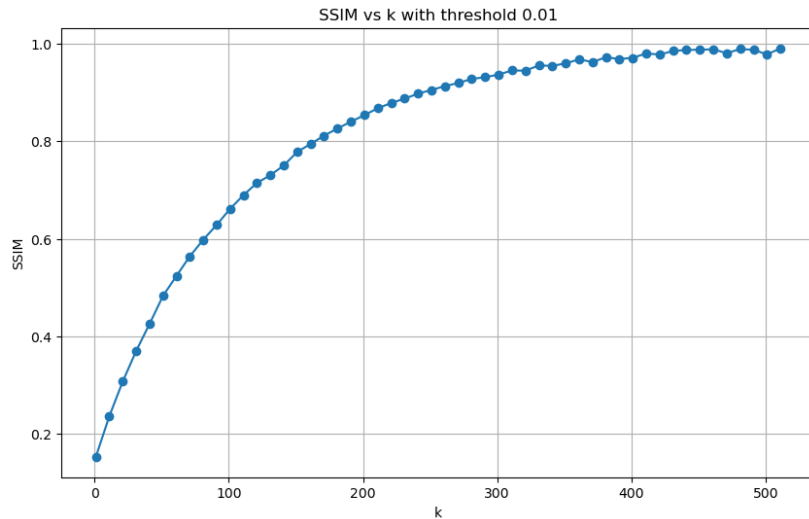


Figure 4.4: SSIM vs k with a threshold of 0.01.

The trends observed in Figure 4.3, Figure 4.4, and Figure 4.5 align with our expectations: as the k value increases, the rank of the low-rank approximation rises, thereby providing a closer approximation to the original matrix. A notable observation from Figure 4.3 is that the graph exhibits fluctuations as the k value approaches its maximum. This behaviour is likely due to information redundancy. Therefore, even minor changes in the interpolative matrix could result in the observed fluctuations in PSNR.

Figure 4.5 illustrates how different k values with the same threshold will affect the compression output. When k is below 50, the compressed image is heavily distorted and almost unrecognisable. However, as the value of k gradually increases to 200, the image shows significant improvement in visual fidelity, with recognisable features becoming clearer. Then, at $k = 200$, the difference between the compressed and original images becomes negligible to the naked eye. This suggests that increasing the value of k enhances the preservation of image details and overall quality in the compressed output.

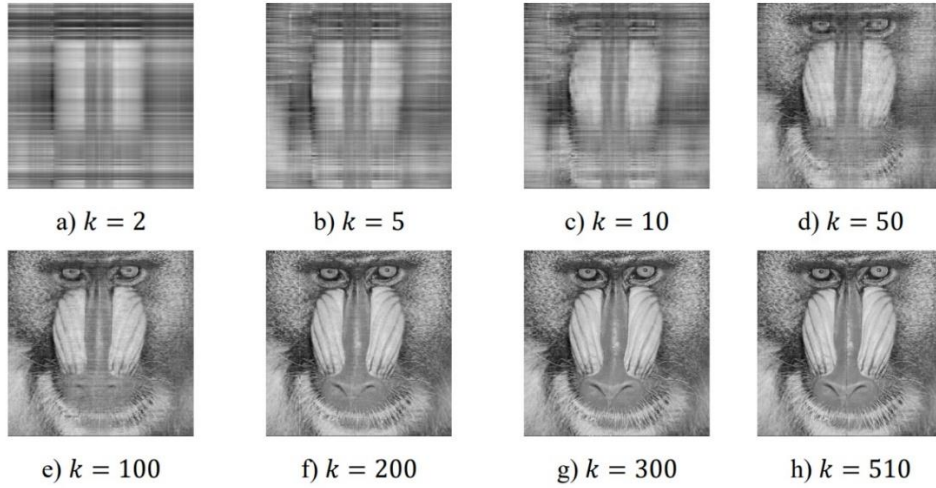


Figure 4.5: ID compression of different k values with a threshold of 0.01.

Table 4.1 lists out the metrics with their respective k values at a threshold of 0.01. From the table, it can be seen that, generally as k increases, the compression ratio decreases.

Table 4.1: Table of k and their respective reconstruction errors and compression ratios.

k	MSE	PSNR	SSIM	Compression Ratio
2	2010.716	15.0973	0.150149	5.803884
5	1463.848	16.4758	0.176223	4.852780
10	1241.537	17.1912	0.226701	4.352396
50	651.644	19.9907	0.477479	3.183527
100	374.1713	22.4001	0.658740	2.808142
200	112.6913	27.6119	0.853026	2.584975
300	42.50894	31.8460	0.940245	2.534132
510	25.88132	34.0009	0.989444	2.536124

Combining the insights from Figure 4.5 and Table 4.1, a k value around 200 appears to offer the best compromise between compression effectiveness and image quality, as the improvement beyond $k = 200$ becomes less noticeable. Additionally, the MSE, PSNR, and SSIM metrics remain at acceptable levels for k around 200.

It can be concluded that there are trade-offs between compression efficiency and image quality at different values of k . Although higher k -values often produce better image quality, they also cause bigger compressed files. As a result, the choice of k depends on the specific needs of the application, such as storage restrictions and the desired level of image fidelity.

4.3 Finding Optimal Parameters

In this section, we determined the optimal parameters for compressing the images of “Baboon”, “Sailboat on Lake”, “Peppers” and “Airplane F-16”. We then derived general parameters which can compress any image without requiring an iterative process.

4.3.1 Optimal Parameters for ID

The optimal compression parameters for the image Baboon, as in Figure 4.1, rank, k and threshold value to compress the image Baboon were determined by iterating the compression process using different k and threshold values. Specifically, we chose 20 equidistant values of k from 1 up to the actual rank of the image. Each k value was paired with threshold values of 0.001, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25 and 0.5. Threshold values exceeding 0.5 were not considered, as the image becomes completely unrecognisable, which does not align with the objective.

Throughout the process, a total of 160 iterations (20 k values \times 8 threshold values) were conducted. For each iteration, metrics such as the Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and compression ratio were recorded and stored in a CSV file for subsequent analysis. A sample of the CSV file for the grayscale Baboon image is shown in TableA-1. SSIM and compression ratio were used to determine the optimal combinations of k and threshold value to compress the image. Figure 4.6 illustrates various compressed Baboon images with their respective SSIM values.

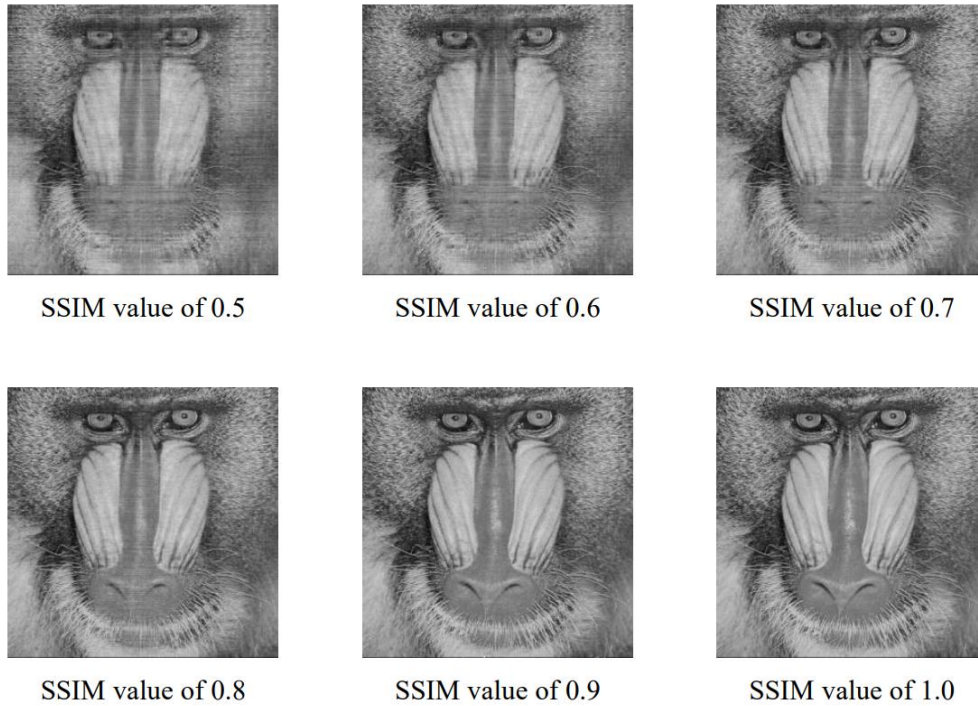
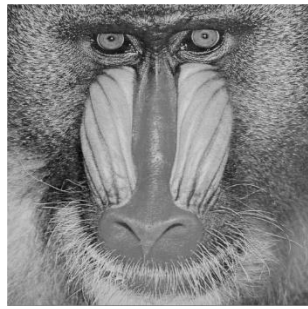


Figure 4.6: Compressed Baboon images with their SSIM values.

From Figure 4.6, it is noticeable that image quality improves as the SSIM value increases. When the SSIM value is below 0.7, the image appears quite blurry; but as it reaches 0.8, the image is visually acceptable. Therefore, an SSIM value of 0.8 was set as the benchmark to deduce the optimal combination. Any combinations resulting in an SSIM value less than 0.8 were discarded. After this filtering based on SSIM, the optimal combination of k and threshold was determined by selecting the configuration with the highest compression ratio from the remaining set of combinations.

Table 4.2: The best configuration results for the Baboon image using ID.

k	Threshold	MSE	PSNR (dB)	SSIM	File Size (bytes)	Compression Ratio
189	0.001	99.559	28.150	0.8413	103208	2.7189
189	0.005	99.559	28.150	0.8413	103208	2.7189
189	0.01	99.559	28.150	0.8413	103208	2.7189
189	0.025	99.559	28.150	0.8413	103208	2.7189



Uncompressed Baboon image

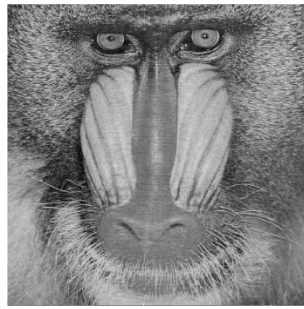
Compressed Baboon image with $k = 189$,
threshold 0.025

Figure 4.7: Comparison between uncompressed and compressed Baboon images with optimal compression parameters.

In Table 4.2, the metrics are the same with varying threshold values. However, we chose the last row with a threshold of 0.025 because a higher threshold would imply more values are being replaced in the interpolation matrix Z . This may aid compression through techniques like Huffman Coding which assigns shorter binary codes to frequently occurring characters.

The results for the Sailboat on Lake image, obtained using a similar process, are presented in Table 4.3, and those for the Peppers image are shown in Table 4.4. Comparative images of the uncompressed and compressed versions for the Sailboat on Lake image and the Peppers image are displayed in Figure 4.8 and Figure 4.9, respectively.

Table 4.3: The best configuration results for the Sailboat on Lake image using ID.

k	Threshold	MSE	PSNR (dB)	SSIM	File Size (bytes)	Compression Ratio
135	0.001	48.799	31.247	0.8395	73694	3.2095
135	0.005	48.799	31.247	0.8395	73694	3.2095
135	0.01	48.799	31.247	0.8395	73694	3.2095



Uncompressed Sailboat on Lake image

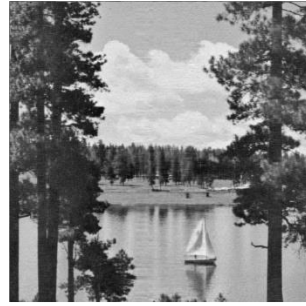
Compressed Sailboat on Lake image with $k = 135$, threshold 0.01

Figure 4.8: Comparison between uncompressed and compressed Sailboat on Lake images with optimal compression parameters.

Table 4.4: The best configuration results for the Peppers image using ID.

k	Threshold	MSE	PSNR (dB)	SSIM	File Size (bytes)	Compression Ratio
135	0.001	32.780	32.975	0.8650	60186	3.5229
135	0.005	32.780	32.975	0.8650	60186	3.5229
135	0.01	32.780	32.975	0.8650	60186	3.5229
135	0.025	32.780	32.975	0.8650	60186	3.5229
135	0.05	32.780	32.975	0.8650	60186	3.5229



Uncompressed Pepper image

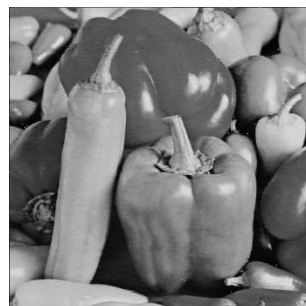
Compressed Pepper image with $k = 135$, threshold 0.05

Figure 4.9: Comparison between uncompressed and compressed Peppers images with optimal compression parameters.

It is observed that the compression ratio is highest for the Peppers image, followed by the Sailboat on Lake image when compared to the Baboon image. This is due to the complexity of the Baboon image, particularly in its detailed fur texture. Images with higher variance in intensity values, like the

Baboon image, typically exhibit lower compression ratios because they contain more unique information that cannot be efficiently compressed. In contrast, the Sailboat on Lake image depicts scenery with similar intensity values throughout, making it easier to compress than the Baboon image. Additionally, the Peppers image, which has the lowest variance of intensity values among the three, achieves the highest compression ratio due to its uniformity in pixel intensity.

The compression ratio alone does not fully indicate the complexity of an image. However, when considered alongside the MSE value, we can identify that the algorithm can compress the Peppers image more effectively compared to the other two. This is because MSE is sensitive to large differences between images, penalising more heavily when the two images differ significantly.

Table 4.5: The best configuration results for the Airplane F-16 image using ID.

k	Threshold	MSE	PSNR (dB)	SSIM	File Size (bytes)	Compression Ratio
457	0.1	0.7554	49.349	0.9963	55160	3.3146

From Table 4.5, we observed that the threshold required for optimal image compression was relatively higher compared to the other images to provide the best results. However, using the same threshold with a different k value produced poor compressed results. Therefore, a threshold value of 0.01 was selected for compressing the Airplane F-16, rather than 0.1. The best result achieved with this threshold value and with an SSIM value greater than 0.8 is shown in Table 4.6. Figure 4.10 displays three images: the uncompressed Airplane F-16 image, the compressed image using the optimal parameter combination, and the compressed image with a threshold value of 0.01.

Table 4.6: The best configuration results for the Airplane F-16 image using ID with a threshold value of 0.01.

k	Threshold	MSE	PSNR (dB)	SSIM	File Size (bytes)	Compression Ratio
511	0.01	0.0010	78.200	0.9999	55163	3.3144

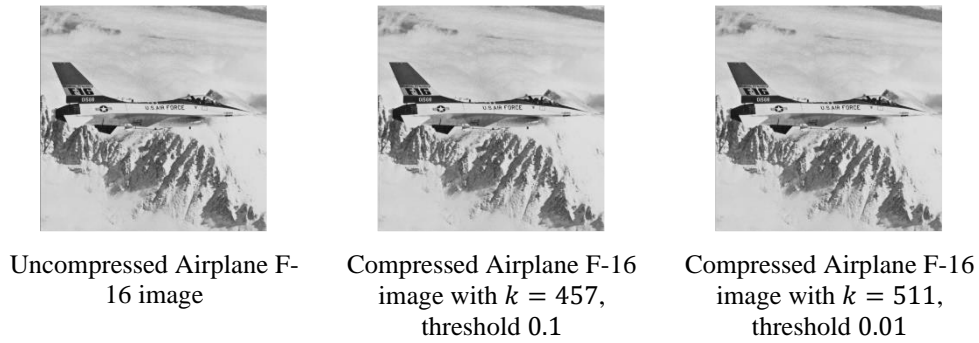


Figure 4.10: Comparison of three images: the uncompressed Airplane F-16 image, the compressed image using the optimal parameters, and the compressed image with a threshold value of 0.01.

In order to generalise a k value for compressing different images, the average k value was calculated as follows:

$$k_{avg} = \frac{189 + 135 + 135 + 511}{4} \approx 243.$$

The four images were recompressed using the average k value and a threshold of 0.01. The results showed that even with this different k value, the outcomes remained acceptable compared to the optimal results obtained from experimentation, as shown in Table 4.7 and Figure 4.11.

Table 4.7: Compression results for the four images using ID with $k_{avg} = 243$ and a threshold of 0.01.

Image Name	Baboon	Sailboat on Lake	Peppers	Airplane F-16
MSE	51.43476	13.74751	10.02563	2.984943
PSNR (dB)	31.01824	36.74856	38.11968	43.38144
SSIM	0.905144	0.939733	0.947379	0.974663
File Size (bytes)	104879	75768	60658	55785
Compression Ratio	2.675579	3.121634	3.495466	3.27744
Difference in Compression Ratio	-0.04332	-0.08785	-0.02741	-0.03696

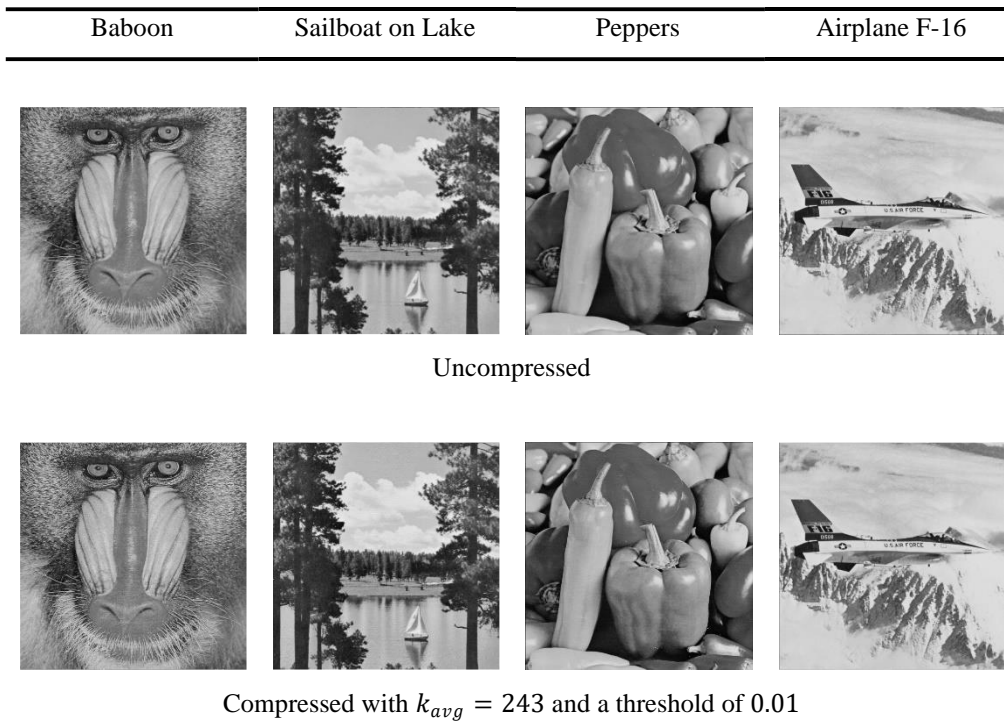


Figure 4.11: Comparative images of their uncompressed and recompressed versions with $k_{avg} = 243$ and a threshold of 0.01.

The decreased compression ratio for the Baboon, Sailboat on Lake, and Peppers images was due to a higher k value chosen to compress the images, indicating that more detailed information was retained, resulting in a lower compression ratio. On the other hand, the Airplane F-16 image, compressed using a smaller k value was expected to yield better compression results according to the earlier logic. However, an anomaly was observed, likely due to the interaction between the nature of ID and the inherent characteristics of the image.

We also attempted to find general compression parameters for 512×512 pixel-coloured images. Unlike grayscale images, colour images have three colour channels: red (R), green (G) and blue (B). Consequently, the colour image was first split into three matrices, each representing one of the three channels. Each matrix then underwent a similar compression process using the same k and threshold value. Finally, all three compressed matrices were combined to produce the final compressed image. Their best configuration results are shown in Table 4.8.

Table 4.8: The best configuration results for the four images using ID.

Image Name	Baboon	Sailboat on Lake	Peppers	Airplane F-16
k	511	162	510	296
Threshold	0.025	0.001	0.01	0.01
MSE	0.000997	72.05922	0.001222	1.896656
PSNR (dB)	78.14481	29.99359	77.55624	45.38779
SSIM	0.999995	0.83849	0.99999	0.980846
File Size (bytes)	122718	88845	73864	63332
Compression Ratio	6.409589	8.853306	10.64892	12.41982

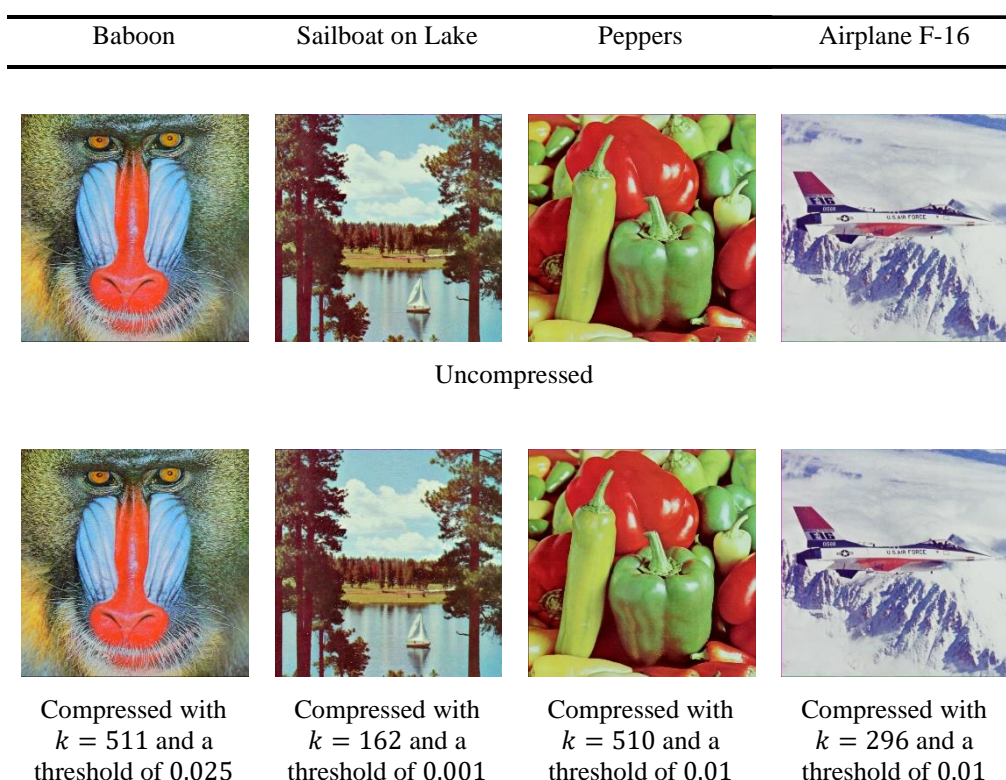


Figure 4.12: Comparative images of uncompressed and optimally compressed versions.

From Table 4.8, we observed that the k value required to compress the Baboon and Peppers images was relatively higher than that of the other two images. As noted earlier, the k value is related to the amount of information retained from the original matrix. A higher k value indicates that more

information was retained. For the Baboon image, the high k value was necessary due to the complexity of the fur texture, similar to the case with its grayscale image.

In contrast, the Peppers image required significantly more k values than its grayscale version due to the vibrant colours of the peppers. In the grayscale version, colour information was discarded, as a result, compressing it will require less information. However, for the colour version, the vibrant colours likely contributed to the need for a higher k value to retain the necessary information for effective compression.

Since more images had 0.01 as their optimal threshold value for compression, we adopted 0.01 as a general threshold for compressing. We then calculated the average k value as follows:

$$k_{avg} = \frac{511 + 162 + 510 + 296}{4} \approx 370.$$

The results for the four images after recompressing with $k_{avg} = 370$ and a threshold of 0.01 is presented in Table 4.9.

Table 4.9: Compression results for the four images using ID with $k_{avg} = 370$ and a threshold of 0.01.

Image Name	Baboon	Sailboat on Lake	Peppers	Airplane F-16
MSE	34.57612	7.379738	200.8264	0.687366
PSNR (dB)	33.70509	40.14759	30.47195	49.79675
SSIM	0.966854	0.977783	0.947861	0.992076
File Size (bytes)	123864	90560	82346	63464
Compression Ratio	6.350287	8.685645	9.552037	12.39399
Difference in Compression Ratio	-0.0593	-0.16766	-1.09688	-0.02583

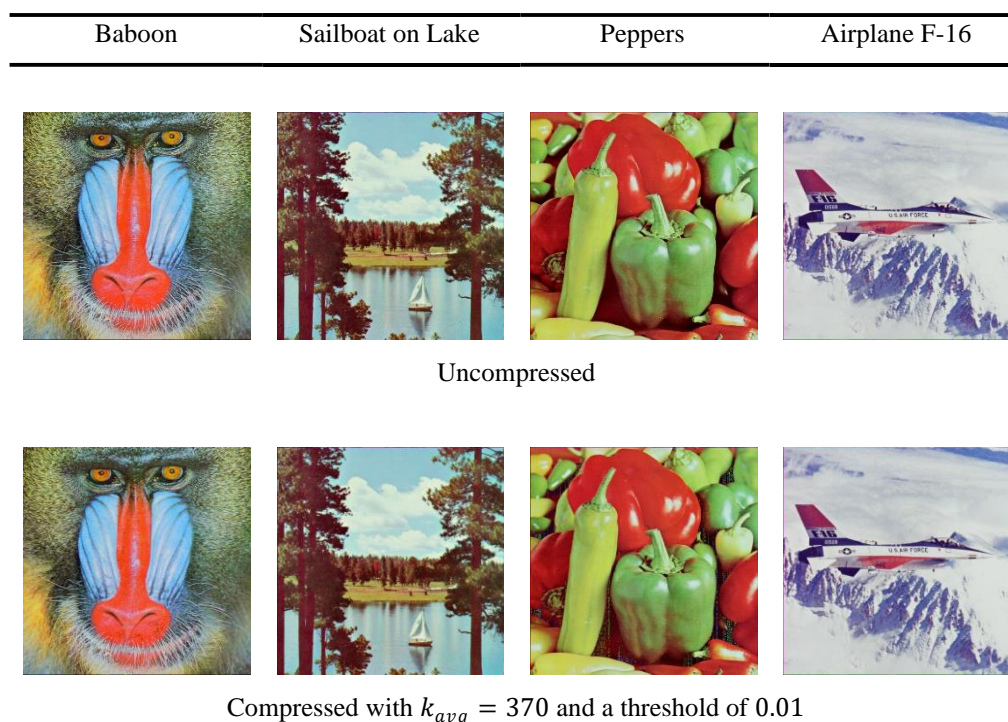


Figure 4.13: Comparative images of their uncompressed and recompressed versions with $k_{avg} = 370$ and a threshold of 0.01.

From Table 4.9, we observed that all four images were recreated with high SSIM values, which indicated good quality. The difference in compression ratios compared to their optimal results was generally acceptable, except for the Sailboat on Lake and Peppers images. The increased difference in compression ratio for the Sailboat on Lake image is due to the difference between the k_{avg} and $k_{Sailboat\ optimal}$, where $k_{Sailboat\ optimal}$ represents the optimal k value for compressing the Sailboat on Lake image.

For the Peppers image, the difference in compression ratio had been significantly larger despite using a smaller k value. By examining the image in Figure 4.13, lines of dots appeared in the darker areas of the image, which explained the high MSE value. This phenomenon resulted in the need for additional information to store the image accurately. The lines of dots were likely caused by the low intensity in those areas, which were consequently discarded due to the small value of k .

The compression parameters were tested on new images to evaluate their performance. The Splash image was used to assess the parameters for both

grayscale and colour images. The results were compared with their optimal values and were listed in Table 4.10.

Table 4.10: Comparison results between compressing using optimal and generalised parameters on both grayscale and colour images using ID.

Image Mode	Grayscale		Colour	
Compression Parameters Used	Optimal	Generalised	Optimal	Generalised
k	55	243	484	370
Threshold	0.01	0.01	0.01	0.01
MSE	30.07837	1.971176	0.027419	159.4838
PSNR (dB)	33.34826	45.18355	68.61479	42.46367
SSIM	0.844145	0.978776	0.999614	0.974872
File Size (bytes)	42574	45729	55700	66448
Compression Ratio	3.922676	3.652037	14.12158	11.83741
Difference in Compression Ratio	–	–0.27064	–	–2.28417

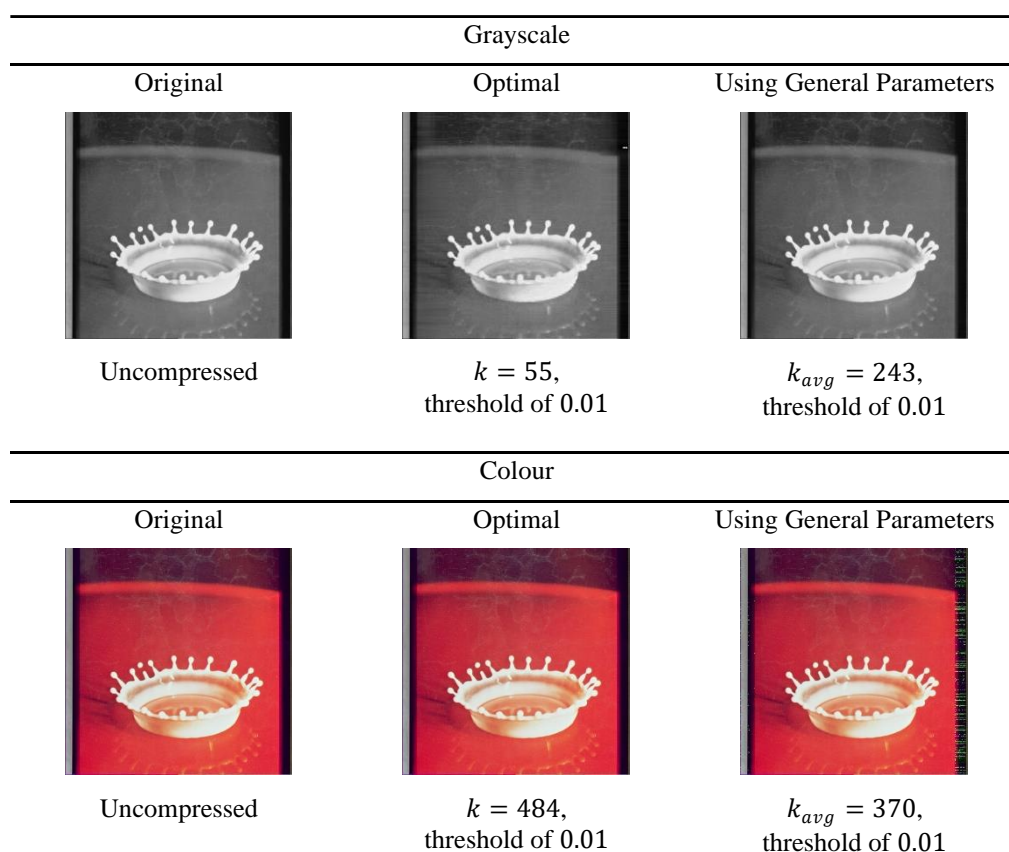


Figure 4.14: Comparative images of uncompressed, optimally compressed, and generally compressed versions.

From Table 4.10 and Figure 4.14, it was evident that the compression results for the tested grayscale image were acceptable. The image quality improved, and the decrease in compression ratio was bearable, even though it used a larger k value. However, the results for the colour version were less satisfactory. Similar issues had arisen when compressing the Peppers image. There were lines of dots in the darker area of the image. This phenomenon indicated that careful consideration is needed when working with images having darker regions, as relying solely on a low-rank approximation can lead to such problems.

In conclusion, for the compression of a 512×512 pixel grayscale 8-bit image using ID, it is recommended to use a k value of 243 along with a threshold value of 0.01. For the colour images of the same resolution, the recommended compression parameters are $k = 370$ and a threshold value of 0.01. However, issues might occur when compressing images around dark areas, such as the Peppers and Splash images.

4.3.2 Optimal Parameters for SVD

The optimal compression parameter for the same set of images using SVD was determined similarly to the method used for ID. The difference between the two methods is that ID includes an additional parameter, the threshold. Therefore, only 20 equidistant values of k from 1 up to the actual rank of the image were iterated to determine the optimal k value. As with the previous method, an SSIM value of 0.8 was used as a benchmark, and any results with an SSIM value less than 0.8 were discarded. Then, the optimal k value was determined by selecting the configuration with the highest compression ratio from the remaining set.

Figure 4.15 illustrates two graphs: (a) the relationship between SSIM and the k value, and (b) the relationship between the compression ratio and k value for the Baboon image. From graph (a), it is evident that the value of k increases along with the SSIM value. Conversely, in graph (b), the relationship between k and the compression ratio is inversely proportional. This phenomenon aligns with the expectation that a higher k value implies less information is being discarded, thus, resulting in a highly similar image (high SSIM) and larger file size (low compression ratio) after compression.

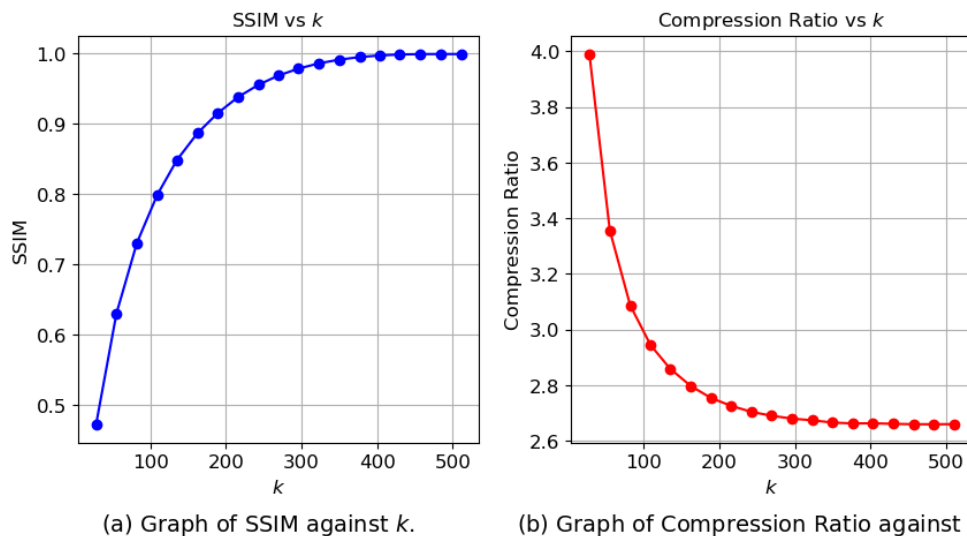


Figure 4.15: Graphs of SSIM and Compression Ratio against k .

The optimal compression result for each image is shown in Table 4.11.

Table 4.11: The best configuration results for the four images using SVD.

Image Name	Baboon	Sailboat on Lake	Peppers	Airplane F-16
k	135	82	82	55
MSE	97.18361	57.0874	48.59727	54.82631
PSNR (dB)	28.25487	30.5654	31.26468	30.74091
SSIM	0.848653	0.821318	0.860182	0.809397
File Size (bytes)	98150	66511	53287	51650
Compression Ratio	2.859012	3.556104	3.978982	3.539826

From Table 4.11, we observed that the Baboon image required a larger k value for compression compared to the others, whereas the Airplane F-16 required the lowest k value. The number of k required is directly linked to the number of information retained. A higher k value is needed for images with higher complexity. When examining the Baboon image, the statement is indeed true, due to its fur texture.

In order to generalise a k value for compressing different images, the average k value was calculated as follows:

$$k_{avg} = \frac{135 + 82 + 82 + 55}{4} \approx 89.$$

The four images were recompressed using the calculated $k_{avg} = 89$. The compression results are listed in Table 4.12.

Table 4.12: Compression results for the four images using SVD with $k_{avg} = 89$.

Image Name	Baboon	Sailboat on Lake	Peppers	Airplane F-16
MSE	188.5694	49.81902	43.8322	21.02483
PSNR (dB)	25.37609	31.15685	31.71287	34.90348
SSIM	0.750196	0.841139	0.871685	0.89274
File Size (bytes)	92348	67665	53912	54571
Compression Ratio	3.038636	3.495456	3.932854	3.350351
Difference in Compression Ratio	0.179625	-0.06065	-0.04613	-0.18947

The comparison of the four images using their best configurations and the average k value ($k_{avg} = 89$) is shown along with their uncompressed versions in Figure 4.16.

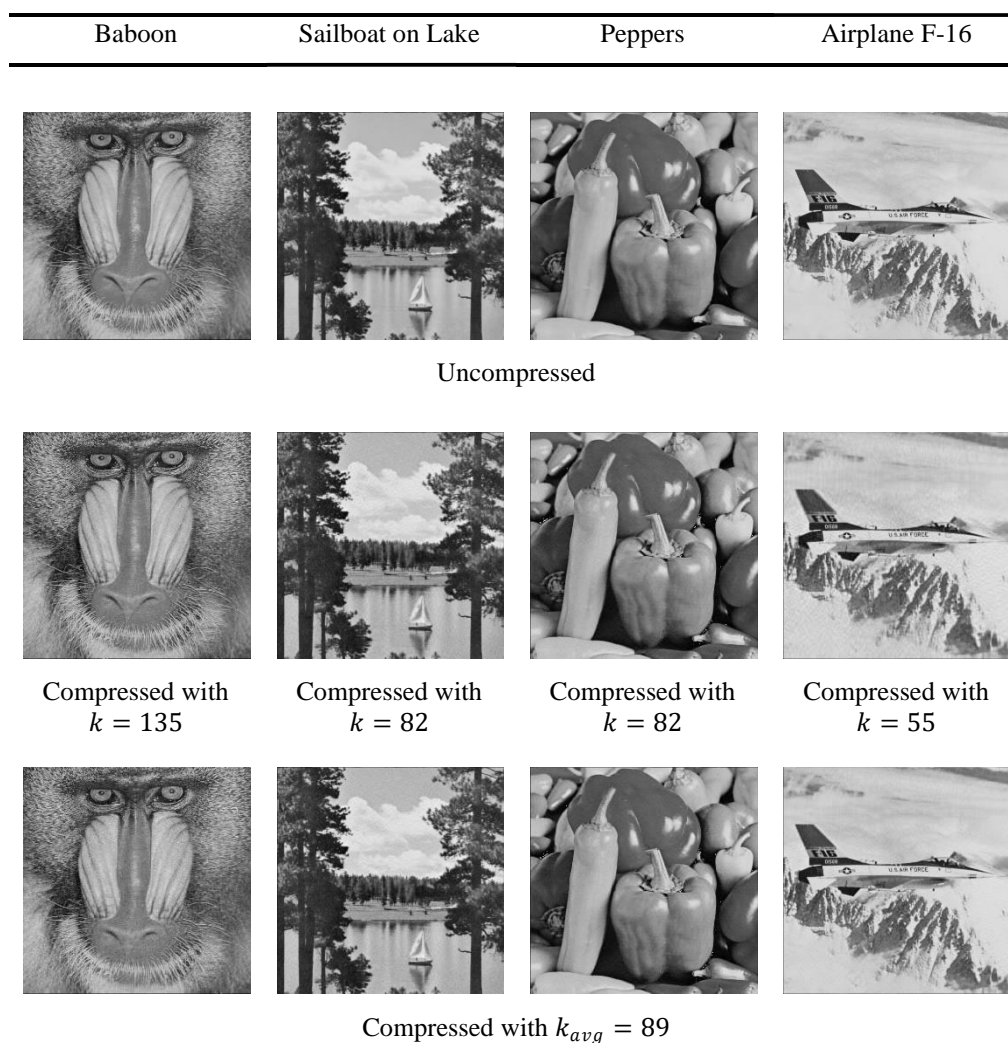


Figure 4.16: Comparison of the four images using their best configurations versus $k_{avg} = 89$, alongside their uncompressed versions.

Due to the nature of SVD, a higher resemblance between the compressed image and the original image (indicated by a higher SSIM) corresponds to a lower compression ratio. Since the Baboon image required the largest k value (compared to the other 3 images) to achieve the optimal result, taking the average of the k values would imply $k_{Baboon\ optimal} < k_{avg}$, where $k_{Baboon\ optimal}$ is the optimal k value for compressing the Baboon image. Consequently, the SSIM value for the Baboon image would be lower than the benchmark value of 0.8.

Additionally, k_{avg} exceeded $k_{Airplane\ F-16\ optimal}$ by more than 50%, where $k_{Airplane\ F-16\ optimal}$ is the optimal k value for compressing the Airplane F-16 image. Compressing the Airplane F-16 using k_{avg} would

involve using more unnecessary information, thereby significantly lowering the compression ratio.

For colour images, the compression process was similar to that used in ID. Each colour image could be split into three layers corresponding to different colour channels. Each of the channels underwent a procedure similar to the grayscale version. Finally, the three compressed channels were recombined to form the final compressed image. The optimal compression results are shown in Table 4.13.

Table 4.13: The best configuration results for the four colour images using SVD.

Image Name	Baboon	Sailboat on Lake	Peppers	Airplane F-16
k	135	89	456	55
MSE	193.5929	101.2867	0.480854	59.52733
PSNR (dB)	25.39016	28.74577	51.31229	30.88547
SSIM	0.831916	0.81573	0.998179	0.813418
File Size (bytes)	117427	83561	73861	61359
Compression Ratio	6.698391	9.413147	10.64935	12.81918

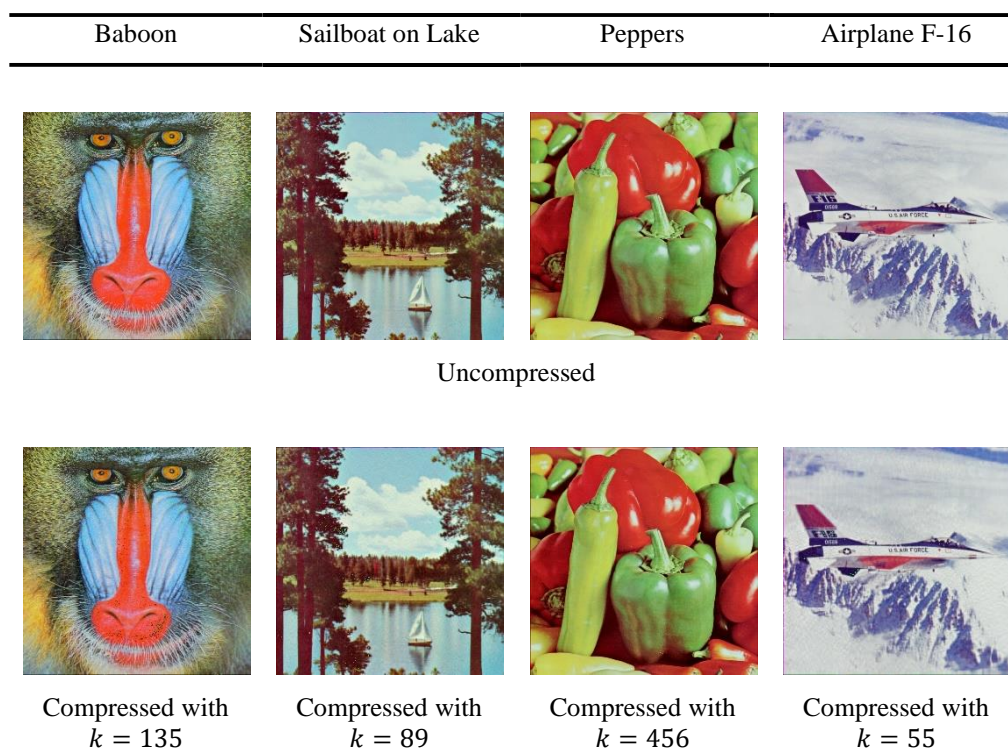


Figure 4.17: Comparative images of uncompressed and optimally compressed versions.

From Table 4.13, we observed an anomaly in the compression results for the Peppers image. Upon examining the compressed image with different k value, we noticed unusual dots appearing around the darker areas. This issue happened likely due to the intensity level in the darker regions are minimal. When performing SVD on these areas, the singular values are also low, which leads to information loss when a lower k value is used for compression. Despite this, we calculate the average k value to identify a general compression parameter:

$$k_{avg} = \frac{135 + 89 + 456 + 55}{4} \approx 184.$$

Table 4.14: Compression results for the four images using SVD with $k_{avg} = 184$.

Image Name	Baboon	Sailboat on Lake	Peppers	Airplane F-16
MSE	110.8986	38.98517	786.3443	16.7668
PSNR (dB)	27.88043	33.20277	24.4599	39.23673
SSIM	0.896224	0.92935	0.877558	0.969713
File Size (bytes)	120135	88266	88782	63621
Compression Ratio	6.547401	8.911382	8.859589	12.3634
Difference in Compression Ratio	-0.15099	-0.50177	-1.78976	-0.45578

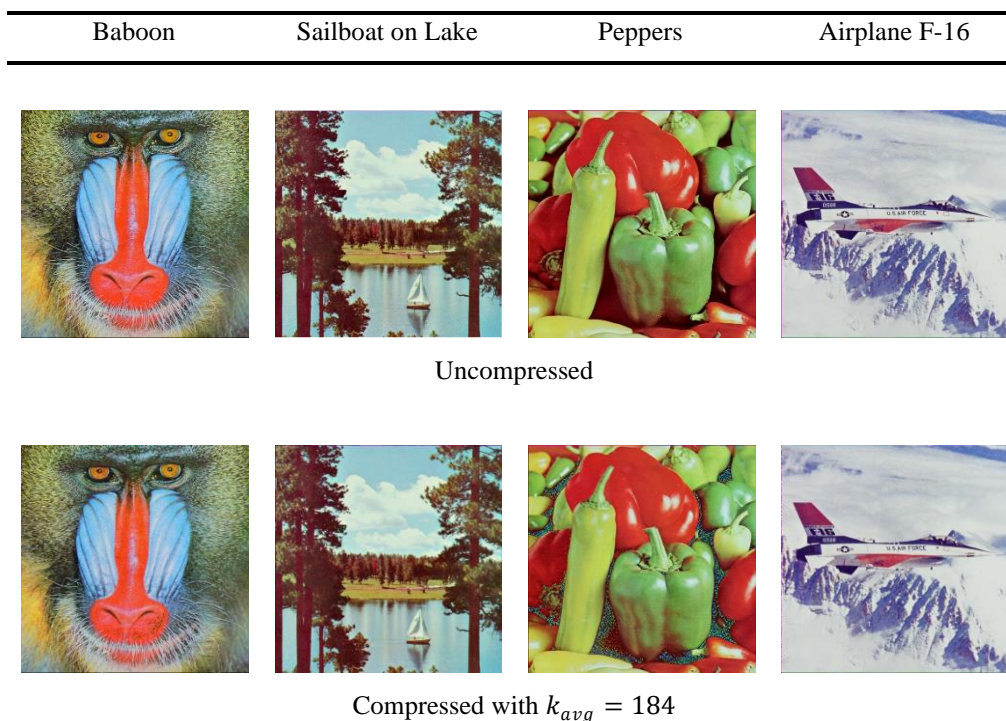


Figure 4.18: Comparative images of their uncompressed and recompressed versions with $k_{avg} = 184$.

According to Table 4.14, the performance for all four images decreased in terms of compression ratio. The reasons for the decrease in performance for the Baboon, Sailboat on Lake, and Airplane F-16 images were similar to those observed with the grayscale version. As a result, approximating the image with

a larger k value required additional storage to capture more details. As for the Peppers image, using a k value smaller than its optimum created dots appearing in the darker areas of the image, which consequently required storing excess information.

The compression parameter was tested on the same image to evaluate their performance. The Splash image was used to assess the parameter for both grayscale and colour images. The results were compared with their optimal values and were listed in.

Table 4.15: Comparison results between compressing using optimal and the general parameter for both grayscale and colour images using SVD.

Image Mode	Grayscale		Colour	
	Optimal	Generalised	Optimal	Generalised
Compression Parameter Used				
k	28	89	28	184
MSE	31.92671	6.080673	451.7643	24.06033
PSNR (dB)	33.08926	40.29129	29.71018	40.92835
SSIM	0.836819	0.937818	0.835039	0.971149
File Size (bytes)	33648	41180	51226	58242
Compression Ratio	4.963267	4.055464	15.35494	13.50524
Difference in Compression Ratio	–	–0.9078	–	–1.8497

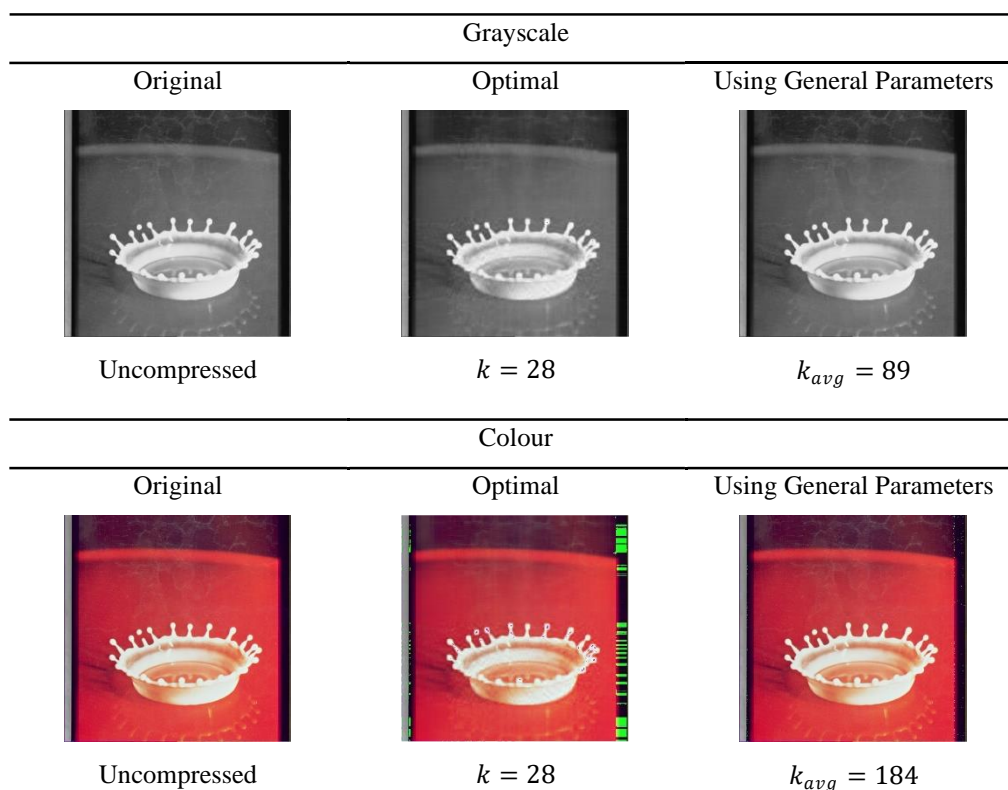


Figure 4.19: Comparative images of uncompressed, optimally compressed, and generally compressed versions.

From Table 4.15 and Figure 4.19, although the difference in compression ratio for both the grayscale and colour images decreased significantly, the comparative image showed improvement. This was due to a higher k value chosen for compression compared to their optimum values. Nevertheless, the final images showed better quality in exchange for a larger file size. For the colour image, the improvements were notable, with the anomaly lines becoming less obvious.

In conclusion, a k value of 89 will be used for compressing 512×512 pixel grayscale 8-bit images using SVD. However, the results may not be consistent due to their image structures and the difference between their optimal compression k value and the average k value of 89. In the case of colour images of the same resolution, the general parameter for compressing was $k = 84$.

4.4 Comparative Analysis of ID-method and SVD-method

In this section, the performance of both the ID and SVD methods was discussed. To compress images, there were two key aspects to consider: maximising the

quality of the final image (represented by SSIM) and minimising the final file size (represented by the compression ratio). However, both aspects could not be achieved simultaneously, as the relationship between them was inversely proportional (the better the image quality, the lower the compression ratio). Therefore, finding a balance between the two was important.

Figure 4.20 presents the comparison graph of SSIM between ID and SVD using the generalised parameters from Section 4.3.1 and 4.3.2, respectively.

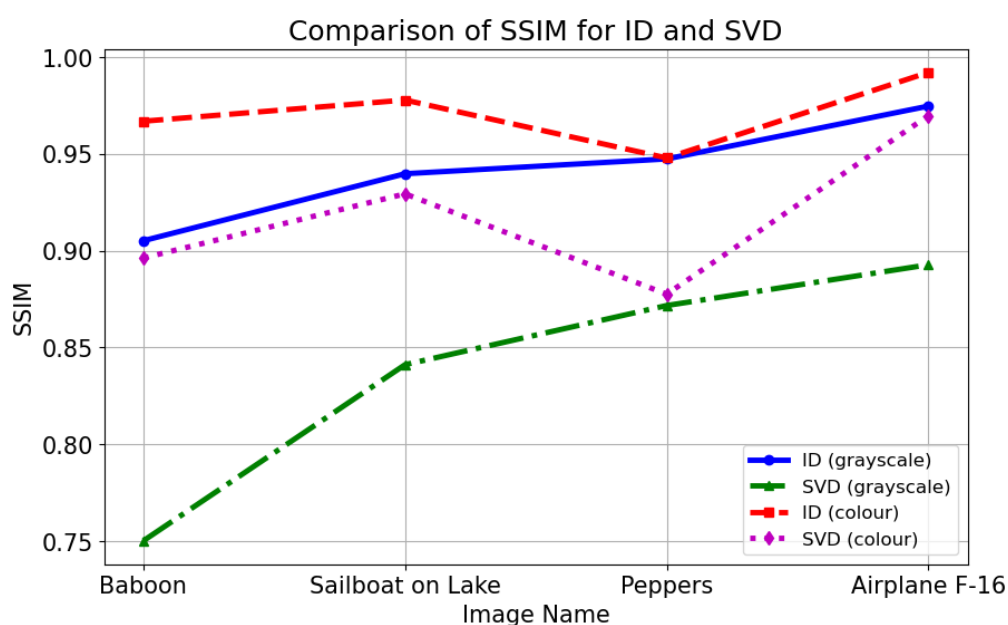


Figure 4.20: Comparison of SSIM for ID and SVD using the generalised parameters.

From Figure 4.20, it was noticeable that when compressing with ID using the generalised parameters, the SSIM was higher compared to using SVD for all four images. As shown in Figure 4.15, the SSIM values increased along with k . Therefore, the optimal parameter would be the minimum k value that achieved at least an SSIM of 0.8 (the benchmark). Consequently, when using the generalised parameter for SVD, the resulting image typically had a relatively lower SSIM value.

Generally, grayscale images had a lower SSIM value compared to their colour counterparts. This is because colour images are separated into three channels, capturing more detailed information. Additionally, since both ID and

SVD compressed the channels separately, more information could be retained from the other channels, thereby increasing the similarity between the original and the compressed images. Therefore, ID would be preferred if maintaining image quality was prioritised over file storage.

Figure 4.21 illustrates the comparison of SSIM for ID and SVD using the generalised parameters.

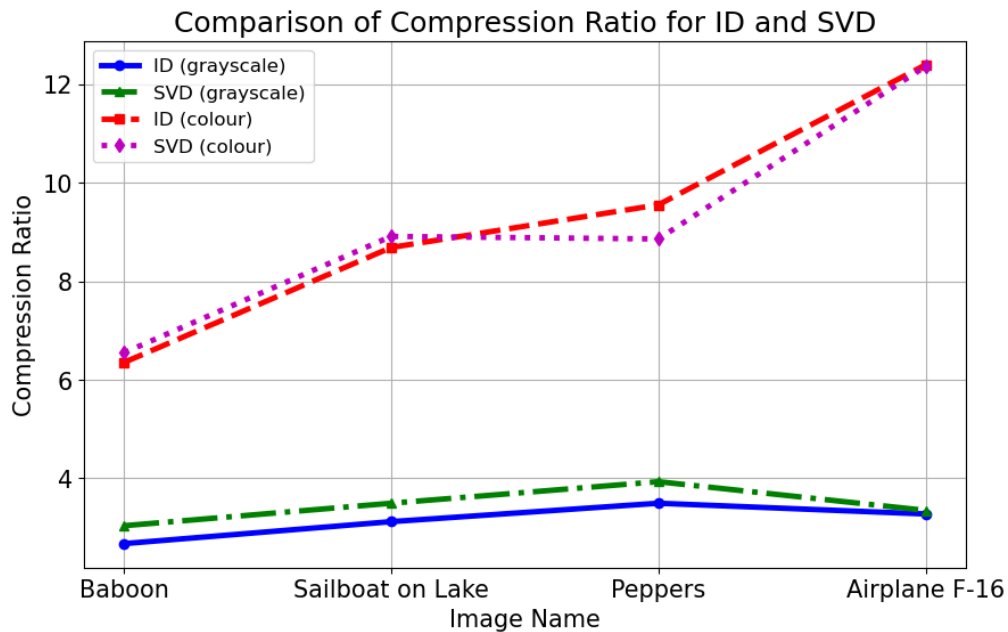


Figure 4.21: Comparison of SSIM for ID and SVD using the generalised parameters.

From Figure 4.21, it is clearly that the graph was divided into two sections: the lower half for grayscale images and the upper half for colour images. The explanation is straightforward, the colour images had more information to store (which represented in channels of colours), while the grayscale images only needed to store information in a 2D array.

Generally, the compression ratio using SVD was higher than that using ID. However, the difference was not significant, indicating that the storage required for the compressed images by these two methods was similar. In the case of larger images, this difference could be further enlarged; therefore, SVD would be preferred if file storage is the sole consideration.

According to the discussions in these sections, the summary is provided in Table 4.16.

Table 4.16: Summary of the similarities and differences between ID and SVD method.

Decomposition Method	ID	SVD
Generalised Parameters	Grayscale: $k = 243$; thresholds of 0.01 Colour: $k = 370$; thresholds of 0.01	Grayscale: $k = 89$ Colour: $k = 184$
Similarities	<ul style="list-style-type: none"> • Required a larger k value to compress complex images. • Not effective at handling dark regions. 	
Pros	Able to retain most information using generalised parameters.	Achieves compression with smaller k values, leading to less storage space requirement.
Cons	Requires a relatively larger k value for compression, resulting in a lower compression ratio.	Dependent on the optimal compression parameter.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In this report, we explored and analysed the applicability of Interpolative Decomposition (ID) in image compression, alongside a comparative analysis with Singular Value Decomposition (SVD). Our investigation aimed to understand the strengths and weaknesses of ID, assess its performance relative to SVD, and identify generalised compression parameters for image compression.

The exploration of ID revealed that it effectively maintained the sparsity of matrices due to its unique properties. Additionally, ID preserved the structure of the matrix by reusing the columns of the original matrix in the approximated low-rank matrix. As a result, it can save both storage space and computational resources. However, it was noted that ID required a relatively larger k value to approximate a dense matrix accurately.

In comparing ID and SVD, we observed similarities in their performance during image compression. Both methods required a larger k value to achieve good compression for complex images, such as the Baboon image, which features dense fur textures. Additionally, both methods struggled to handle colour images containing dark regions. Consequently, requiring a higher k value to address the issue. In terms of their advantages, SVD showed superiority when the primary goal was to optimise storage, whereas ID proved advantageous when the quality of the compressed image was more important. This comparison emphasised the importance of selecting the appropriate method based on the specific objective, such as final image size or desired quality.

We also determined generalised compression parameters for compressing images using ID and SVD. These parameters were tested with a sample image, yielding ideal results. For a 512×512 pixel grayscale image, the general compression parameters were found to be $k = 243$ with a threshold of 0.01 for ID, and $k = 89$ for SVD. For colour images of the same resolution,

the parameters were $k = 370$ with a threshold of 0.01 for ID, and $k = 184$ for SVD.

5.2 Recommendations for future work

The technique used to manipulate the interpolation matrix Z in the compression of ID was relatively straightforward. Future research could focus on refining this technique and exploring additional methods to enhance image processing capabilities. For instance, rather than compressing the entire matrix directly, one could partition the matrix into smaller blocks and compress each block separately before combining them to form the final compressed image. However, this method might require additional time for compression in exchange for better image quality.

Furthermore, the algorithm could be improved by selecting a suitable set of numbers to be used in the interpolation matrix Z , and then replacing values with those closest to the numbers in this predefined list. By doing so, the matrix would consist of a limited range of values, allowing for a storage method that efficiently handles frequently occurring numbers. This method theoretically could lead to a more optimised image file size by focusing on common values.

REFERENCES

- Advani, R. and O'Hagan, S., 2022. Efficient algorithms for constructing an interpolative decomposition. *arXiv*.
- Advani, R., Crim, M. and O'Hagan, S., 2020. Random projections and dimension reduction. *arXiv*.
- Anderson, E., Bai, Z. and Dongarra, J., 1992. Generalized QR factorization and its applications. *Linear Algebra and its Applications*, 162, pp.243-271.
- Bentbib, A.H., Kreit, K. and Labaali, I., 2022. *Randomized tensor singular value decomposition for multidimensional data compression*. In: IEEE, 11th International Symposium on Signal, Image, Video and Communications (ISIVC). El Jadida, Morocco, May 2022.
- Bhaskara, A., Lattanzi, S., Vassilvitskii, S. and Zadimoghaddam, M., 2020. *Residual based sampling for online low rank approximation*. In: IEEE, 2020 Information Theory and Applications Workshop (ITA). San Diego, CA, USA, February 2022.
- Damle, A., Lin, L. and Ying, L., 2017. SCDM-k: Localized orbitals for solids via selected columns of the density matrix. *Journal of Computational Physics*, 334, pp.1-15.
- Golub, G., 1965. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7(3), pp.206-216.
- Golub, G.H. and Van Loan, C.F., 2013. *Matrix computations*. 4th ed. Baltimore: The Johns Hopkins University Press.
- Intawichai, S. and Chaturantabut, S., 2022. A missing data reconstruction method using an accelerated least-squares approximation with randomized SVD. *Algorithms*, 15(6), pp.190.
- Kawamura, H. and Suda, R., 2021. Least upper bound of truncation error of low-rank matrix approximation algorithm using QR decomposition with pivoting. *Japan Journal of Industrial and Applied Mathematics*, 38(3), pp.757-779.
- Kiran, Parameshachari, B.D., Kumar, D.S., Prafulla, P.S. and Yashwanth, J., 2023. *Singular Value Decomposition (SVD) based optimal image compression technique*. In: IEEE, 2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT). Bengaluru, India, October 2023.
- Kong, S., Sun, L., Han, C. and Guo, J., 2017. An image compression scheme in wireless multimedia sensor networks based on NMF. *Information*, 8(1), pp.26.

- Lay, D.C., Lay, S.R. and McDonald, J.J., 2014. *Linear algebra and its applications*. 5th ed. Pearson.
- Lee, D. and Seung, H.S., 2000. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13. MIT Press.
- Libal, U., Baras, J.S. and Johansson, K.H., 2020. *Dimensionality reduction of volterra kernels by tensor decomposition using higher-order SVD*. In: IEEE, 59th IEEE Conference on Decision and Control (CDC). Jeju, Korea (South), December 2020.
- Liberty, E., Woolfe, F., Martinsson, P.G., Rokhlin, V. and Tygert, M., 2007. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51), pp.20167-20172.
- Lin, C.J., 2007. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10), pp.2756-2779.
- Liu, W. and He, M., 2019. Accelerating solution of volume-surface integral equations with multiple right-hand sides by improved skeletonization techniques. *IEEE Antennas and Wireless Propagation Letters*, 18(10), pp.2006-2010.
- Lu, J. and Ying, L., 2015. Compression of the electron repulsion integral tensor in tensor hypercontraction format with cubic scaling cost. *Journal of Computational Physics*, 302, pp.329-335.
- Mersereau, R.M., 1979. The processing of hexagonally sampled two-dimensional signals. *Proceedings of the IEEE*, 67(6), pp.930-949.
- Mittal, R.C. and Al-Kurdi, A., 2002. Efficient solution of a sparse non-symmetric system of linear equations. *International journal of computer mathematics*, 79(4), pp.449-463.
- Muravev, A., Tran, D.T., Iosifidis, A., Kiranyaz, S. and Gabbouj, M., 2018. *Acceleration approaches for big data analysis*. In: IEEE, 25th IEEE International Conference on Image Processing (ICIP). Athens, Greece, October 2018.
- Sahu, K.K. and Satao, P.K.J., 2016. Image compression methods using dimension reduction and classification through PCA and LDA: A review. *International Journal of Science and Research*, 5(5), pp.2277-2280.
- Strang, G., 2006. *Linear algebra and its applications*. 4th ed. Boston: Cengage Learning.
- Su, Q., Wang, G., Zhang, X., Lv, G. and Chen, B., 2018. A new algorithm of blind color image watermarking based on LU decomposition. *Multidimensional Systems and Signal Processing*, 29(3), pp.1055-1074.

Tang, W.K.A., Ng, W.S. and Liew, H.H., 2023. Separation of two musical instruments using matrix factorisation techniques. *International Journal of Applied Mathematics*, 36(3), pp.425.

Tornberg, A.K., 2013. *Linear algebra, part 3, QR and SVD*. [pdf] Royal Institute of Technology. Available at: <https://www.math.kth.se/na/DN2266/mod1-13/LA3_2p.pdf> [Accessed 26 August 2024]

Varghese, P. and Saroja, G.A.S., 2021. *Hexagonal image compression using singular value decomposition in Python*. In: IEEE, 2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS 2021). Ernakulam, India, 2-4 September 2021.

Woolfe, F., Liberty, E., Rokhlin, V. and Tygert, M., 2008. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3), pp.335-366.

Wüthrich, C.A. and Stucki, P., 1991. An algorithmic comparison between square-and hexagonal-based grids. *CVGIP: Graphical Models and Image Processing*, 53(4), pp.324-339.

Ying, L., Damle, A., Lin, L. and Lu, J., 2018. *Interpolative decomposition and its applications in quantum chemistry*. In: CSCAMM, 2018 Mathematical and Numerical Aspects of Quantum Dynamics. Maryland, United States, 19-21 June 2018.

Zhang, R., 2024. *Matrix decomposition series: 4 — principles and applications of Non-negative Matrix Factorization (NMF)*. [online] Available at: <<https://rendazhang.medium.com/matrix-decomposition-series-4-principles-and-applications-of-non-negative-matrix-factorization-4361b55f9c9e>> [Accessed 3 August 2024].

APPENDICES

Appendix A: Tables

TableA-1: Sample of Results Stored In A CSV File For The Grayscale
Baboon Image Compressed Using ID.

k	threshold	mse	psnr	ssim	file_size	compression_ratio
2	0.001	1650.104	15.95569	0.150196	46576	6.02482
28	0.001	744.3648	19.41295	0.350787	74486	3.767312
55	0.001	502.7615	21.11718	0.50018	85430	3.284701
82	0.001	366.8684	22.4857	0.602505	91983	3.050694
109	0.001	266.5912	23.87235	0.686399	96535	2.906842
135	0.001	196.3527	25.20043	0.747024	99710	2.814281
162	0.001	138.5697	26.71412	0.799559	101757	2.757668
189	0.001	99.55919	28.14999	0.841333	103208	2.718898
216	0.001	71.73163	29.5737	0.875785	103953	2.699412
243	0.001	51.41388	31.02	0.905176	104853	2.676242
269	0.001	37.95246	32.3384	0.925151	105359	2.663389
296	0.001	26.87882	33.8367	0.942068	105680	2.655299
323	0.001	19.00764	35.34152	0.95762	105881	2.650258
350	0.001	13.79339	36.73409	0.970333	105869	2.650559
377	0.001	9.120949	38.5304	0.98042	105769	2.653065
403	0.001	6.641659	39.90804	0.987404	105699	2.654822
430	0.001	3.306774	42.93676	0.992921	105593	2.657487
457	0.001	2.142971	44.82064	0.997257	105535	2.658947
484	0.001	0.888279	48.64531	0.999311	105440	2.661343
511	0.001	0.000984	78.20001	0.999996	105401	2.662328
2	0.005	1650.104	15.95569	0.150196	46576	6.02482
28	0.005	744.3674	19.41293	0.350784	74481	3.767565
55	0.005	502.7615	21.11718	0.50018	85430	3.284701
82	0.005	366.8684	22.4857	0.602505	91983	3.050694
109	0.005	266.5912	23.87235	0.686399	96535	2.906842
135	0.005	196.3527	25.20043	0.747024	99710	2.814281
162	0.005	138.5697	26.71412	0.799559	101757	2.757668
189	0.005	99.55919	28.14999	0.841333	103208	2.718898
216	0.005	71.73429	29.57354	0.875787	103968	2.699023
243	0.005	51.41665	31.01977	0.90517	104862	2.676012
269	0.005	37.95246	32.3384	0.925151	105359	2.663389
296	0.005	26.87882	33.8367	0.942068	105680	2.655299
323	0.005	19.00764	35.34152	0.95762	105881	2.650258
350	0.005	13.79391	36.73393	0.970331	105887	2.650108
377	0.005	9.120949	38.5304	0.98042	105769	2.653065
403	0.005	6.641659	39.90804	0.987404	105699	2.654822
430	0.005	3.306774	42.93676	0.992921	105593	2.657487
457	0.005	2.146797	44.81289	0.997252	105542	2.658771

484	0.005	0.888279	48.64531	0.999311	105440	2.661343
511	0.005	0.000984	78.20001	0.999996	105401	2.662328
2	0.01	1650.104	15.95569	0.150196	46576	6.02482
28	0.01	744.3674	19.41293	0.350784	74481	3.767565
55	0.01	502.7615	21.11718	0.50018	85430	3.284701
82	0.01	366.8746	22.48563	0.602494	91983	3.050694
109	0.01	266.5912	23.87235	0.686399	96535	2.906842
135	0.01	196.3527	25.20043	0.747024	99710	2.814281
162	0.01	138.5777	26.71387	0.799554	101763	2.757505
189	0.01	99.55919	28.14999	0.841333	103208	2.718898
216	0.01	71.73987	29.5732	0.875784	103965	2.699101
243	0.01	51.43476	31.01824	0.905144	104879	2.675579
269	0.01	37.95246	32.3384	0.925151	105359	2.663389
296	0.01	26.87882	33.8367	0.942068	105680	2.655299
323	0.01	19.00764	35.34152	0.95762	105881	2.650258
350	0.01	13.79391	36.73393	0.970331	105887	2.650108
377	0.01	9.120949	38.5304	0.98042	105769	2.653065
403	0.01	6.641659	39.90804	0.987404	105699	2.654822
430	0.01	3.306774	42.93676	0.992921	105593	2.657487
457	0.01	2.146797	44.81289	0.997252	105542	2.658771
484	0.01	0.895958	48.60793	0.999282	105430	2.661595
511	0.01	0.000984	78.20001	0.999996	105401	2.662328
2	0.025	1650.16	15.95554	0.150112	46712	6.007279
28	0.025	744.4177	19.41264	0.350741	74492	3.767009
55	0.025	502.9038	21.11595	0.500138	85476	3.282933
82	0.025	367.0244	22.48385	0.60227	92017	3.049567
109	0.025	266.619	23.87189	0.686349	96540	2.906692
135	0.025	196.3527	25.20043	0.747024	99710	2.814281
162	0.025	139.2025	26.69433	0.79927	101786	2.756882
189	0.025	99.55919	28.14999	0.841333	103208	2.718898
216	0.025	71.73987	29.5732	0.875784	103965	2.699101
243	0.025	51.80497	30.98709	0.904777	104875	2.675681
269	0.025	37.95246	32.3384	0.925151	105359	2.663389
296	0.025	26.91368	33.83107	0.942031	105672	2.6555
323	0.025	19.00764	35.34152	0.95762	105881	2.650258
350	0.025	13.84145	36.71899	0.970188	105886	2.650133
377	0.025	9.120949	38.5304	0.98042	105769	2.653065
403	0.025	6.641659	39.90804	0.987404	105699	2.654822
430	0.025	3.366608	42.85888	0.992699	105583	2.657738
457	0.025	2.195656	44.71516	0.997087	105547	2.658645
484	0.025	0.895958	48.60793	0.999282	105430	2.661595
511	0.025	0.000984	78.20001	0.999996	105401	2.662328
2	0.05	1650.869	15.95368	0.149618	47005	5.969833
28	0.05	745.5494	19.40604	0.35	74749	3.754057
55	0.05	504.5117	21.10209	0.498991	85688	3.274811
82	0.05	370.1935	22.44652	0.600122	92409	3.036631
109	0.05	267.4439	23.85848	0.685504	96644	2.903564

135	0.05	197.5139	25.17483	0.746093	99773	2.812504
162	0.05	142.2337	26.60078	0.796597	101962	2.752123
189	0.05	100.5745	28.10593	0.839795	103340	2.715425
216	0.05	72.38821	29.53413	0.874667	104075	2.696248
243	0.05	54.65582	30.75444	0.902188	105046	2.671325
269	0.05	38.00722	32.33214	0.925112	105368	2.663161
296	0.05	27.04153	33.81049	0.941819	105663	2.655726
323	0.05	19.55145	35.21901	0.956633	105868	2.650584
350	0.05	14.53925	36.50538	0.968719	105942	2.648732
377	0.05	9.342354	38.42624	0.980218	105790	2.652538
403	0.05	6.641659	39.90804	0.987404	105699	2.654822
430	0.05	3.366608	42.85888	0.992699	105583	2.657738
457	0.05	3.459385	42.74081	0.99364	105706	2.654646
484	0.05	1.455086	46.50192	0.997616	105500	2.659829
511	0.05	0.237541	54.37341	0.999188	105469	2.660611
2	0.1	1653.874	15.94578	0.149051	47729	5.879277
28	0.1	939.0886	18.40374	0.305924	83086	3.377368
55	0.1	858.0319	18.79577	0.425502	95423	2.940717
82	0.1	744.3473	19.41305	0.504999	101064	2.776577
109	0.1	384.3778	22.28322	0.635107	100716	2.786171
135	0.1	300.1633	23.35723	0.700343	103191	2.719346
162	0.1	505.028	21.09765	0.658269	108918	2.57636
189	0.1	119.3873	27.36122	0.821606	104290	2.690689
216	0.1	121.1155	27.29881	0.834734	105888	2.650083
243	0.1	199.0125	25.142	0.805017	109426	2.5644
269	0.1	46.30955	31.4741	0.913099	105944	2.648682
296	0.1	34.75857	32.72018	0.931361	106159	2.643318
323	0.1	33.66245	32.85935	0.939209	106585	2.632753
350	0.1	26.73894	33.85936	0.951669	106758	2.628487
377	0.1	15.97456	36.09651	0.968991	106231	2.641526
403	0.1	9.686218	38.26926	0.982737	105857	2.650859
430	0.1	29.19716	33.4774	0.959171	106879	2.625511
457	0.1	102.2901	28.03247	0.900499	108894	2.576928
484	0.1	28.85151	33.52912	0.966379	106655	2.631025
511	0.1	10.46948	37.93155	0.992565	105808	2.652087
2	0.25	1756.226	15.685	0.146007	48851	5.744243
28	0.25	7210.931	9.55089	0.03218	165267	1.697931
55	0.25	7301.682	9.496575	0.010792	179123	1.566588
82	0.25	7245.564	9.530082	0.012344	183123	1.532369
109	0.25	7150.497	9.587441	0.01311	183849	1.526318
135	0.25	7259.887	9.521505	0.006504	184384	1.521889
162	0.25	7284.138	9.507022	0.010512	184348	1.522186
189	0.25	4541.479	11.55883	0.133667	175972	1.59464
216	0.25	7121.643	9.605002	0.015105	184320	1.522418
243	0.25	7231.765	9.538361	0.01444	184608	1.520042
269	0.25	3345.662	12.88598	0.246844	167241	1.67789
296	0.25	3000.133	13.3594	0.300045	163333	1.718036

323	0.25	2660.589	13.88103	0.354178	160210	1.751526
350	0.25	2268.937	14.57258	0.414505	155144	1.80872
377	0.25	1886.888	15.37334	0.47164	149415	1.878071
403	0.25	1476.863	16.4374	0.562522	137253	2.044487
430	0.25	1136.906	17.57356	0.644696	136316	2.05854
457	0.25	782.6599	19.19507	0.729898	130198	2.155271
484	0.25	415.5972	21.94408	0.838113	119278	2.352588
511	0.25	15.14066	36.32936	0.991363	106215	2.641924
2	0.5	5183.541	10.98454	0.130474	58727	4.778245
28	0.5	7621.699	9.310286	0.033997	111607	2.514287
55	0.5	7174.352	9.572977	0.020412	131872	2.127912
82	0.5	7224.019	9.543015	0.019887	147368	1.904158
109	0.5	7072.019	9.635369	0.022937	157801	1.778265
135	0.5	7370.224	9.455997	0.00823	164090	1.71011
162	0.5	7340.775	9.473384	0.018218	172286	1.628757
189	0.5	7326.364	9.481919	0.012593	176093	1.593544
216	0.5	7132.542	9.59836	0.017428	178051	1.57602
243	0.5	7229.549	9.539692	0.015026	181447	1.546523
269	0.5	7373.169	9.454262	0.00673	181967	1.542104
296	0.5	7386.319	9.446523	0.008155	181866	1.54296
323	0.5	7033.146	9.659307	0.022039	180505	1.554594
350	0.5	7093.487	9.622206	0.017308	180184	1.557364
377	0.5	7114.038	9.609642	0.027638	179947	1.559415
403	0.5	7169.393	9.575979	0.019442	180030	1.558696
430	0.5	7215.429	9.548182	0.020755	177916	1.577216
457	0.5	7132.059	9.598654	0.023292	175397	1.599868
484	0.5	7337.9	9.475086	0.017086	175412	1.599731
511	0.5	7486.23	9.388172	0.015723	173394	1.618349